PrimeRail
Tool Commands

Version M-2017.06-SP2, September 2017

SYNOPSYS®

Copyright Notice and Proprietary Information

Â© 2017 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at
http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

add_to_collection . . . 17
alias . . . 20
all_clocks . . . 22
all_connected . . . 23
all_fanin . . . 25
all_fanout . . . 28
all_inputs . . . 31
all_instances . . . 33
all_outputs . . . 35
all_registers . . . 37
append_to_collection . . . 41
apropos . . . 43
associate_supply_set . . . 45
calculate_effective_resistance . . . 47
calculate_minimum_path_resistances . . . 49
calculate_rail_voltage . . . 51
calculate_rush_currents . . . 57
cell_of . . . 61
change_macro_view . . . 62
change_net_type . . . 64
change_pin_type . . . 66
change_selection . . . 68
characterize_context . . . 71
check_ccs_power . . . 73
check_constraints . . . 75
check_level_shifters . . . 77
check_mw_lib . . . 78
check_noise . . . 80
check_power . . . 83
check_supply_net_integrity . . . 85
check_taps . . . 94
check_timing . . . 96
close_mw_cel . . . 102
close_mw_lib . . . 104
collections . . . 105
compare_collections . . . 109
compare_interface_timing . . . 111
complete_net_parasitics . . . 117
connect_pg_net . . . 119
connect_supply_net . . . 121
convert_mw_lib . . . 123
copy_collection . . . 125

copy_mw_cel 127
copy_mw_lib 130
cputime 132
create_cell 133
create_clock 137
create_command_group 140
create_generated_clock 142
create_mtcmos_switch_cell_sequence 147
create_mw_cel 150
create_mw_lib 152
create_net 154
create_operating_conditions 157
create_pg_pin_only_lib_cell 160
create_placement_blockage 162
create_power_rail_mapping 164
create_power_switch 167
create_qtm_constraint_arc 170
create_qtm_delay_arc 173
create_qtm_drive_type 176
create_qtm_generated_clock 179
create_qtm_load_type 181
create_qtm_model 183
create_qtm_path_type 185
create_qtm_port 188
create_rail_macro_model 190
create_si_context 199
create_supply_net 202
create_supply_port 204
create_taps 206
create_voltage_area 212
current_design 215
current_instance 217
current_integrity_result 220
current_mw_cel 222
current_mw_lib 224
current_rail_result 226
date 229
define_design_mode_group 231
define_name_maps 233
define_proc_attributes 236
define_qtm_attribute 241
define_scaling_lib_group 243
define_user_attribute 245
derive_clocks 248
disconnect_net 250
drive_of 253
echo 255

enable_primetime_icc_consistency_settings . . . 257
error_info . . . 258
estimate_clock_network_power . . . 260
exit . . . 263
export_power_management_mtcmos_locations . . . 265
extract_model . . . 267
extract_supply_parasitics . . . 271
filter_collection . . . 273
find . . . 276
find_objects . . . 278
foreach_in_collection . . . 281
generate_rail_macro_model_probes . . . 283
generate_reduced_core_model . . . 285
get_alternative_lib_cells . . . 288
get_app_var . . . 291
get_attribute . . . 294
get_cells . . . 296
get_clock_network_objects . . . 299
get_clocks . . . 301
get_command_option_values . . . 303
get_core_area . . . 305
get_current_power_domain . . . 307
get_current_power_net . . . 309
get_defined_commands . . . 311
get_designs . . . 314
get_die_area . . . 317
get_em_max_toggle_rate . . . 319
get_generated_clocks . . . 321
get_latch_loop_groups . . . 323
get_layers . . . 325
get_lib_cells . . . 328
get_lib_pins . . . 331
get_lib_timing_arcs . . . 334
get_libs . . . 337
get_license . . . 340
get_macro_current_sources . . . 342
get_macro_mtcmos . . . 345
get_message_ids . . . 348
get_message_info . . . 350
get_net_shapes . . . 352
get_nets . . . 356
get_noise_violation_sources . . . 360
get_object_name . . . 363
get_path_groups . . . 365
get_pg_pins . . . 367
get_pin_shapes . . . 370
get_pins . . . 374

get_ports . . . 377
get_power_domains . . . 380
get_power_group_objects . . . 382
get_power_switches . . . 384
get_qtm_ports . . . 386
get_random_numbers . . . 388
get_selection . . . 390
get_si_bottleneck_nets . . . 393
get_supply_nets . . . 395
get_supply_ports . . . 398
get_supply_sets . . . 400
get_switching_activity . . . 402
get_taps . . . 409
get_terminals . . . 413
get_timing_arcs . . . 417
get_timing_paths . . . 421
get_user_shapes . . . 432
get_vias . . . 436
getenv . . . 439
group_path . . . 441
gui_start . . . 445
gui_stop . . . 447
help . . . 448
history . . . 450
identify_interface_logic . . . 453
import_power_management_mtcmos_locations . . . 457
index_collection . . . 459
infer_switching_activity . . . 461
insert_buffer . . . 463
insert_virtual_decap_cells . . . 471
is_false . . . 474
is_true . . . 476
license_users . . . 478
link_design . . . 480
list_attributes . . . 484
list_designs . . . 487
list_key_bindings . . . 489
list_libs . . . 490
list_licenses . . . 492
list_mw_cels . . . 494
list_sidefiles . . . 496
lminus . . . 498
load_of . . . 500
load_power_management_mtcmos_locations . . . 501
ls . . . 503
man . . . 504
map_design_mode . . . 506

mem . . . 508
merge_saif . . . 509
open_mw_cel . . . 512
open_mw_lib . . . 514
parse_proc_arguments . . . 516
print_message_info . . . 518
print_suppressed_messages . . . 520
printenv . . . 521
printvar . . . 523
proc_args . . . 525
proc_body . . . 527
query_cell_instances . . . 529
query_cell_mapped . . . 530
query_net_ports . . . 531
query_objects . . . 533
query_port_net . . . 536
quit . . . 537
read_db . . . 538
read_def . . . 540
read_em_rules . . . 543
read_file . . . 546
read_lef . . . 548
read_lib . . . 551
read_parasitics . . . 553
read_saif . . . 559
read_sdc . . . 562
read_sdf . . . 567
read_supply_parasitics . . . 571
read_vcd . . . 573
read_verilog . . . 577
redirect . . . 580
remove_annotated_check . . . 584
remove_annotated_clock_network_power . . . 588
remove_annotated_delay . . . 590
remove_annotated_mtcmos_switch_cell . . . 592
remove_annotated_parasitics . . . 595
remove_annotated_power . . . 597
remove_annotated_transition . . . 599
remove_buffer . . . 601
remove_capacitance . . . 604
remove_case_analysis . . . 606
remove_cell . . . 608
remove_clock . . . 611
remove_clock_gating_check . . . 613
remove_clock_groups . . . 615
remove_clock_latency . . . 617
remove_clock_sense . . . 619

remove_clock_transition 620
remove_clock_uncertainty 622
remove_connection_class 625
remove_context 626
remove_coupling_separation 628
remove_data_check 630
remove_design_mode 632
remove_disable_clock_gating_check 634
remove_disable_timing 636
remove_drive_resistance 638
remove_driving_cell 640
remove_fanout_load 642
remove_from_collection 644
remove_generated_clock 646
remove_host_options 648
remove_ideal_latency 653
remove_ideal_network 655
remove_ideal_transition 657
remove_input_delay 659
remove_input_noise 661
remove_integrity_result 663
remove_intrinsic_rc 665
remove_lib 667
remove_license 669
remove_max_area 671
remove_max_capacitance 673
remove_max_fanout 675
remove_max_time_borrow 677
remove_max_transition 679
remove_min_capacitance 681
remove_min_pulse_width 683
remove_mtcmos_switch_cell_sequence 685
remove_mw_cel 687
remove_net 690
remove_noise_immunity_curve 692
remove_noise_lib_pin 694
remove_noise_margin 696
remove_operating_conditions 698
remove_output_delay 699
remove_parasitic_corner 701
remove_path_group 702
remove_port_fanout_number 704
remove_power_groups 706
remove_power_management_mtcmos_locations 708
remove_probes 710
remove_propagated_clock 712
remove_pulse_clock_max_transition 714

remove_pulse_clock_max_width . . . 716
remove_pulse_clock_min_transition . . . 718
remove_pulse_clock_min_width . . . 720
remove_qtm_attribute . . . 722
remove_rail_current . . . 724
remove_rail_macro_mode_sequence . . . 726
remove_rail_maps . . . 728
remove_rail_power . . . 729
remove_rail_result . . . 731
remove_rail_voltage . . . 734
remove_resistance . . . 736
remove_sense . . . 738
remove_setup_hold_pessimism_reduction . . . 740
remove_si_aggressor_exclusion . . . 742
remove_si_delay_analysis . . . 744
remove_si_delay_disable_statistical . . . 747
remove_si_noise_analysis . . . 749
remove_si_noise_disable_statistical . . . 752
remove_steady_state_resistance . . . 754
remove_taps . . . 756
remove_user_attribute . . . 758
remove_virtual_decap_cells . . . 760
remove_wire_load_min_block_size . . . 762
remove_wire_load_model . . . 763
remove_wire_load_selection_group . . . 765
rename . . . 767
rename_cell . . . 769
rename_design . . . 772
rename_integrity_result . . . 774
rename_mw_lib . . . 777
rename_net . . . 779
rename_rail_result . . . 782
report_activity_file_check . . . 785
report_activity_waveforms . . . 790
report_alternative_lib_cells . . . 792
report_analysis_coverage . . . 796
report_annotated_check . . . 800
report_annotated_delay . . . 803
report_annotated_mtcmos_switch_cell . . . 806
report_annotated_parasitics . . . 809
report_annotated_power . . . 813
report_app_var . . . 815
report_attribute . . . 817
report_bottleneck . . . 820
report_bus . . . 824
report_case_analysis . . . 826
report_cell . . . 829

report_cell_usage . . . 833
report_clock . . . 836
report_clock_gate_savings . . . 841
report_clock_gating_check . . . 847
report_clock_timing . . . 849
report_constraint . . . 860
report_context . . . 871
report_crpr . . . 874
report_current_calculation . . . 880
report_delay_calculation . . . 886
report_design . . . 896
report_design_mismatch . . . 898
report_disable_timing . . . 900
report_driver_model . . . 903
report_em_options . . . 906
report_em_rules . . . 908
report_em_violations . . . 910
report_etm_arc . . . 915
report_exceptions . . . 919
report_extract_supply_parasitics_options . . . 924
report_extraction_hierarchy . . . 926
report_global_slack . . . 928
report_global_timing . . . 931
report_hard_macro_power_management_mtcmos . . . 937
report_hierarchy . . . 939
report_host_usage . . . 941
report_ideal_network . . . 945
report_integrity_results . . . 948
report_latch_loop_groups . . . 951
report_lib . . . 953
report_lib_groups . . . 959
report_min_period . . . 961
report_min_pulse_width . . . 965
report_mode . . . 969
report_mtcmos_switch_cell_sequence . . . 973
report_mtcmos_switch_lib_cell . . . 976
report_mw_lib . . . 978
report_name_mapping . . . 980
report_net . . . 982
report_noise . . . 985
report_noise_calculation . . . 988
report_noise_parameters . . . 992
report_noise_violation_sources . . . 994
report_ocvm . . . 997
report_path_group . . . 1000
report_pg_pin_mapping . . . 1002
report_pg_pin_only_lib_cell . . . 1004

report_port 1006
report_power 1009
report_power_analysis_options 1022
report_power_calculation 1024
report_power_derate 1030
report_power_domain 1033
report_power_groups 1036
report_power_pin_info 1038
report_power_rail_mapping 1041
report_power_switch 1043
report_probes 1045
report_pulse_clock_max_transition 1047
report_pulse_clock_max_width 1050
report_pulse_clock_min_transition 1053
report_pulse_clock_min_width 1055
report_qor 1057
report_qtm_model 1061
report_rail_checks 1064
report_rail_current 1067
report_rail_macro_model 1069
report_rail_power 1076
report_rail_results 1078
report_rail_voltage 1080
report_reference 1083
report_scale_parasitics 1085
report_sense 1087
report_si_aggressor_exclusion 1089
report_si_bottleneck 1092
report_si_delay_analysis 1096
report_si_double_switching 1101
report_si_noise_analysis 1103
report_spice_options 1108
report_supply_net_hierarchy 1110
report_supply_network 1112
report_supply_parasitics 1115
report_supply_set 1117
report_supply_voltage 1119
report_switched_net_state 1121
report_switching_activity 1123
report_taps 1132
report_threshold_voltage_group 1136
report_timing 1139
report_timing_derate 1158
report_transitive_fanin 1162
report_transitive_fanout 1165
report_units 1168
report_vcd_hierarchy 1170

report_virtual_decap_cells 1172
report_voltage_regulator_lib_cell 1174
report_wire_load 1176
reset_mode 1178
reset_noise_parameters 1181
reset_path 1182
reset_power_derate 1186
reset_rtl_to_gate_name 1188
reset_scale_parasitics 1189
reset_switching_activity 1191
reset_timing_derate 1195
retrieve_rail_result 1198
save_mw_cel 1200
save_qtm_model 1204
scale_parasitics 1206
scale_rail_power 1208
set_active_clocks 1211
set_annotated_check 1213
set_annotated_clock_network_power 1217
set_annotated_delay 1220
set_annotated_mtcmos_switch_cell 1224
set_annotated_power 1228
set_annotated_transition 1231
set_app_var 1233
set_case_analysis 1235
set_clock_gating_check 1238
set_clock_groups 1241
set_clock_latency 1245
set_clock_map 1250
set_clock_sense 1255
set_clock_transition 1256
set_clock_uncertainty 1258
set_connection_class 1262
set_context_margin 1264
set_coupling_separation 1267
set_cross_voltage_domain_analysis_guardband 1269
set_data_check 1271
set_design_attributes 1274
set_design_top 1276
set_disable_clock_gating_check 1277
set_disable_timing 1279
set_distributed_parameters 1283
set_domain_supply_net 1285
set_dont_override 1287
set_dont_touch 1289
set_dont_touch_network 1292
set_dont_use 1294

set_drive 1296
set_drive_resistance 1299
set_driving_cell 1301
set_effective_voltage_options 1304
set_em_options 1306
set_equal 1308
set_extract_supply_parasitics_options 1310
set_false_path 1313
set_fanout_load 1317
set_host_options 1319
set_ideal_latency 1321
set_ideal_network 1323
set_ideal_transition 1325
set_input_delay 1327
set_input_noise 1332
set_input_transition 1334
set_intrinsic_rc 1336
set_isolation 1338
set_isolation_control 1342
set_latch_loop_breaker 1344
set_level_shifter_strategy 1346
set_level_shifter_threshold 1348
set_lib_rail_connection 1350
set_library_driver_waveform 1352
set_load 1354
set_max_area 1358
set_max_capacitance 1360
set_max_delay 1363
set_max_fanout 1368
set_max_time_borrow 1370
set_max_transition 1372
set_message_info 1374
set_min_capacitance 1376
set_min_delay 1378
set_min_library 1382
set_min_pulse_width 1384
set_mode 1387
set_mtcmos_switch_lib_cell 1390
set_multicycle_path 1393
set_mw_lib_reference 1399
set_mw_lib_write_access 1401
set_noise_derate 1403
set_noise_immunity_curve 1406
set_noise_lib_pin 1409
set_noise_margin 1411
set_noise_parameters 1413
set_operating_conditions 1415

set_opposite 1419
set_output_delay 1421
set_parasitic_corner 1426
set_pg_pin_mapping 1428
set_port_attributes 1430
set_port_fanout_number 1433
set_power_clock_scaling 1435
set_power_derate 1437
set_probes 1440
set_program_options 1442
set_propagated_clock 1444
set_pulse_clock_max_transition 1446
set_pulse_clock_max_width 1448
set_pulse_clock_min_transition 1450
set_pulse_clock_min_width 1452
set_qtm_attribute 1454
set_qtm_global_parameter 1456
set_qtm_port_drive 1458
set_qtm_port_load 1461
set_qtm_technology 1463
set_query_rules 1466
set_rail_current 1470
set_rail_macro_mode_sequence 1474
set_rail_macro_model_cells 1476
set_rail_power 1478
set_related_supply_net 1482
set_resistance 1484
set_retention 1486
set_retention_control 1489
set_rtl_to_gate_name 1491
set_scope 1493
set_sense 1495
set_setup_hold_pessimism_reduction 1499
set_si_aggressor_exclusion 1501
set_si_delay_analysis 1503
set_si_delay_disable_statistical 1506
set_si_noise_analysis 1508
set_si_noise_disable_statistical 1511
set_spice_options 1513
set_steady_state_resistance 1516
set_supply_voltage 1518
set_switched_net_state 1520
set_switching_activity 1522
set_tap_package_connection 1527
set_tap_package_model 1529
set_temperature 1534
set_timing_derate 1536

set_units 1543
set_user_attribute 1545
set_user_units 1547
set_vectorfree_rail_analysis_options 1549
set_voltage 1551
set_voltage_regulator_lib_cell 1553
set_well_capacitance 1555
set_wire_load_min_block_size 1557
set_wire_load_mode 1559
set_wire_load_model 1561
set_wire_load_selection_group 1564
setenv 1566
sh 1568
size_cell 1570
sizeof_collection 1574
sort_collection 1576
source 1578
start_hosts 1580
start_icc2_gui 1583
start_profile 1585
stop_hosts 1587
stop_profile 1592
suppress_message 1594
swap_cell 1596
transform_exceptions 1600
unalias 1605
unset_rtl_to_gate_name 1606
unsetenv 1608
unsuppress_message 1610
update_currents 1612
update_em 1615
which 1616
write_activity_waveforms 1618
write_app_var 1622
write_arrival_annotations 1624
write_changes 1626
write_context 1631
write_interface_timing 1634
write_parasitics 1637
write_physical_annotations 1639
write_rail_data 1642
write_rlib_template 1645
write_saif 1647
write_script 1650
write_sdc 1652
write_sdf 1656
write_spice_deck 1662

write_taps 1668

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection add_to_collection
   [-unique]
   collection1
   object_spec
```

Data Types

collection1	collection
object_spec	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

collection1

Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. The *collection1* option can be the empty collection (empty string), subject to some constraints, as explained in the DESCRIPTION section.

object_spec

Specifies a list of named objects or collections to add.

If the base collection is heterogeneous, only collections can be added to it.

If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

The *object_spec* has some special rules when the base collection is empty, as explained in the DESCRIPTION section.

DESCRIPTION

The **add_to_collection** command allows you to add elements to a collection. The result is a new collection representing the objects in the *object_spec* added to the objects in the base collection.

Elements that exist in both the base collection and the *object_spec*, are duplicated in the resulting collection. Duplicates are not removed unless you use the *-unique* option. If the *object_spec* is empty, the result is a copy of the base collection.

If the base collection is homogeneous, the command searches in the database for any elements of the *object_spec* that are not collections, using the object class of the base collection. If the base collection is heterogeneous, all implicit elements of the *object_spec* are ignored.

When the *collection1* argument is the empty collection, some special rules apply to the *object_spec*. If the *object_spec* is non-empty, there must be at least one homogeneous collection somewhere in the *object_spec* list (its position in the list does not matter). The first homogeneous collection in the *object_spec* list becomes the base collection and sets the object class for the function. The examples show the different errors and warnings that can be generated.

The **append_to_collection** command has similar semantics as the **add_to_collection** command; however, the **append_to_collection** command can be much more efficient in some cases. For more information about the command, see the man page.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime uses the **get_ports** command to get all of the ports beginning with 'mode' and then adds the "CLOCK" port.

```
pt_shell> set xports [get_ports mode*]
{"mode[0]", "mode[1]", "mode[2]"}
pt_shell> add_to_collection $xports [get_ports CLOCK]
{"mode[0]", "mode[1]", "mode[2]", "CLOCK"}
```

The following example from PrimeTime adds the cell u1 to a collection containing the SCANOUT port.

```
pt_shell> set so [get_ports SCANOUT]
{"SCANOUT"}
pt_shell> set u1 [get_cells u1]
{"u1"}
pt_shell> set het [add_to_collection $so $u1]
{"u1"}
pt_shell> query_objects -verbose $het
{"port:SCANOUT", "cell:u1"}
```

The following examples show how the **add_to_collection** command behaves when the base collection is empty. Adding two empty collections yields the empty collection. Adding an implicit list of only strings or heterogeneous collections to the empty collection generates an error message, because no homogeneous collections are present in the *object_spec* list. Finally, as long as one homogeneous collection is present in the *object_spec* list, the command succeeds, even though a warning message is generated. The example uses the variable settings from the previous example.

```
pt_shell> sizeof_collection [add_to_collection "" ""]
0

pt_shell> set A [add_to_collection "" [list a $het c]]
Error: At least one homogeneous collection required for argument 'object_spec'
        to add_to_collection when the 'collection' argument is empty (SEL-014)

pt_shell> add_to_collection "" [list a $het $sp]]
Warning: Ignored all implicit elements in argument 'object_spec'
         to add_to_collection because the class of the base collection
         could not be determined (SEL-015)
{"SCANOUT", "u1", "SCANOUT"}
```

SEE ALSO

append_to_collection(2)
collections(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)

alias

Creates a pseudo-command that expands to one or more words, or lists current alias definitions.

SYNTAX

```
string alias
[name]
[def]
```

Data Types

```
name      string
def       string
```

ARGUMENTS

name

Specifies a name of the alias to define or display. The name must begin with a letter, and can contain letters, underscores, and numbers.

def

Expands the alias. That is, the replacement text for the alias name.

DESCRIPTION

The **alias** command defines or displays command aliases. With no arguments, the **alias** command displays all currently defined aliases and their expansions. With a single argument, the **alias** command displays the expansion for the given alias name. With more than one argument, an alias is created that is named by the first argument and expanding to the remaining arguments.

You cannot create an alias using the name of any existing command or procedure. Thus, you cannot use **alias** to redefine existing commands.

Aliases can refer to other aliases.

Aliases are only expanded when they are the first word in a command.

EXAMPLES

Although commands can be abbreviated, sometimes there is a conflict with another command. The following example shows how to use **alias** to get around the conflict:

```
prompt> alias q quit
```

The following example shows how to use **alias** to create a shortcut for commonly-used command invocations:

```
prompt> alias include {source -echo -verbose}

prompt> alias rt100 {report_timing -max_paths 100}
```

After the previous commands, the command **include script.tcl** is replaced with **source -echo -verbose script.tcl** before the command is interpreted.

The following examples show how to display aliases using **alias**. Note that when displaying all aliases, they are in alphabetical order.

```
prompt> alias rt100
rt100  report_timing -max_paths 100

prompt> alias
include  source -echo -verbose
q  quit
rt100  report_timing -max_paths 100
```

SEE ALSO

unalias(2)

all_clocks

Creates a collection of all clocks in the current design. You can assign these clocks to a variable or pass them into another command.

SYNTAX

```
collection all_clocks
```

ARGUMENTS

None.

DESCRIPTION

The **all_clocks** command creates a collection of all clocks in the current design. If you do not define any clocks, the empty collection (empty string) is returned.

If you want only certain clocks, use **get_clocks** to create a collection of clocks matching a specific pattern and optionally pass in filter criteria.

EXAMPLES

The following example applies the **set_propagated_clock** command to all clocks in the design.

```
pt_shell> set_propagated_clock [all_clocks]
```

SEE ALSO

```
collections(2)
create_clock(2)
derive_clocks(2)
get_clocks(2)
set_propagated_clock(2)
```

all_connected

Creates a collection of objects connected to a net, pin, or port object. You can assign this collection to a variable or pass it into another command.

SYNTAX

```
collection all_connected
   [-leaf]
   object_spec
```

Data Types

```
object_spec       list
```

ARGUMENTS

-leaf

Specifies that the connections of the net that are being returned should be global or leaf pins. When specified, this gives the leaf pins of a hierarchical net. For non-hierarchical nets, there is no difference in output.

object_spec

Specifies the object whose connections are returned. This is a collection of one element which is a net, pin, or port collection, or the name of a net, pin, or port.

DESCRIPTION

The **all_connected** command creates a collection of objects connected to a specified net, pin, or port. The *object_spec* option is either a collection of exactly one net, pin, or port object, or a name of an object. If it is a name, PrimeTime searches for a net, pin, or port, in that order. If the *object_spec* refers to a net, you can use the *-leaf* option to get the global leaf pins of the net.

The command returns a collection. The collection can contain nets, ports, pins, or a combination of ports and pins. In the latter case, the ports are first and they are followed by the pins.

When issued from the command prompt, the **all_connected** command behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example shows all objects connected to net "CLOCK":

```
pt_shell> query_objects -verbose [all_connected [get_nets CLOCK]]
{"port:CLOCK", "pin:U1/CP", "pin:U2/CP", "pin:U3/CP", "pin:U4/CP"}
```

SEE ALSO

collections(2)
get_nets(2)
get_pins(2)
query_objects(2)
collection_result_display_limit(3)

all_fanin

Creates a collection of pins, ports, or cells in the fanin of specified objects.

SYNTAX

```
collection all_fanin
   [-from from_list]
   [-through through_list]
   -to to_list
   [-startpoints_only]
   [-only_cells]
   [-flat]
   [-quiet]
   [-step_into_hierarchy]
   [-levels level_count]
   [-pin_levels pin_count]
   [-trace_arcs timing | enabled | all]
   [-continue_trace pin_types]
```

Data Types

```
from_list      list
through_list   list
to_list        list
level_count    int
pin_count      int
pin_types       string
```

ARGUMENTS

-from *from_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each object in *to_list* becomes part of the resulting collection only if it is in the fanout cone of at least one object in *from_list*. If a net is specified, the effect is the same as listing all load pins on the net.

-through *through_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified the fanin of each object in *to_list* is restricted to objects on paths through the pins, ports or nets in *through_list*.

-to *to_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each object in *to_list* becomes part of the resulting collection. If a net is

specified, the effect is the same as listing all driver pins on the net. This argument is required.

-startpoints_only

Includes only the timing startpoints in the result.

-only_cells

Includes only cells in the timing fanin of the *to_list* and not pins or ports.

-flat

There are two major modes in which **all_fanin** functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current to object are included in the result. In flat mode, the only non-leaf objects in the result are hierarchical to pins.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-step_into_hierarchy

You can only use this option in hierarchical mode and with either the **-levels** or **-pin_levels** option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the input pins are considered a single level away from the related output pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related output pins.

-levels *level_count*

The traversal stops when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the object in *to_list*.

-pin_levels *pin_count*

The traversal stops when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the object in *to_list*.

-trace_arcs timing | enabled | all

Specifies the type of combinational arcs to trace during the traversal:

- **timing** (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- **enabled** - Permits the tracing of all enabled arcs and disregards case analysis values.
- **all** - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

-continue_trace *pin_types*

This option permits the user to override the default behavior of the traversal which is to stop at all timing startpoints. When used, the travsal is permitted through startpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- **generated_clock_source** - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

DESCRIPTION

The **all_fanin** command creates a collection of objects in the timing fanin of specified to pins/ports or nets in the design. A pin is considered to be in the timing fanin of an object in *to_list* if there is a timing path through combinational logic from the pin to that object (see also the **-trace_arcs** option). The fanin stops at the clock pins of registers (sequential cells).

If a current instance in the design is not the top level of hierarchy, only objects within the current instance are returned.

The fanin cone can be topologically restricted by using the **-from** and **-through** options.

EXAMPLES

This example shows the timing fanin of a port in the design. The fanin includes a register, reg1.

```
pt_shell> query_objects [all_fanin -to out_1]
{"out_1", "reg1/Q", "reg1/CP"}
```

This example shows the flat mode of **all_fanin**. The object in *to_list* is an input pin of a hierarchical cell, H1, which is connected to an output pin of another hierarchical cell, H2. H2 contains additional hierarchy and eventually, a leaf cell with 2 inputs, each of which has a top level register in its fanin.

```
pt_shell> query_objects [all_fanin -to H1/a -flat]
{"H1/a", "H2/U1/n1/Z", "H2/U1/n1/A", "H2/U1/n1/B",
 "reg1/Q", "reg2/Q", "reg1/CP", "reg2/CP"}
```

This example shows the timing fanin of an output port that passes through pin U1/Z or pin U2/Z.

```
pt_shell> query_objects [all_fanin -to out -through {U1/Z U2/Z}]
{"out", "U3/Z", "U3/A", "U3/B", "U1/Z", "U1/A", "U2/Z", "U2/Z",
 "reg1/Q", "reg2/Q", "reg1/CP", "reg2/CP"}
```

This example shows the timing fanin of an output port the passes through pin U1/Z and pin U3/A.

```
pt_shell> query_objects [all_fanin -to out -through U1/Z -through U3/A]
{"out", "U3/Z", "U3/A", "U1/Z", "U1/A", "reg1/Q", "reg1/CP"}
```

SEE ALSO

```
all_fanout(2)
current_instance(2)
report_transitive_fanin(2)
report_transitive_fanout(2)
```

all_fanout

Creates a collection of pins, ports, or cells in the fanout of the specified objects.

SYNTAX

```
collection all_fanout
   -from from_list
   [-through through_list]
   [-to to_list]
   -clock_tree
   [-endpoints_only]
   [-only_cells]
   [-flat]
   [-quiet]
   [-step_into_hierarchy]
   [-levels level_count]
   [-pin_levels pin_count]
   [-trace_arcs timing | enabled | all]
   [-continue_trace pin_types]
```

Data Types

```
from_list         list
through_list      list
to_list           list
level_count       int
pin_count         int
pin_types         string
```

ARGUMENTS

-from *from_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each object in *from_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all load pins on the net. This option is exclusive with the **-clock_tree** option.

-through *through_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified the fanout of each object in *from_list* is restricted to pins on paths through the pins, ports or nets in *through_list*.

-to *to_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each object in *from_list* becomes part of the resulting collection if it is in

the fanin cone of at least one object in *to_list*. If a net is specified, the effect is the same as listing all driver pins on the net.

-clock_tree

Specifies that all clock source pins and ports in the design are used as objects in the *from_list*. Clock sources are specified using **create_clock**. If there are no clocks, or if the clocks have no sources, the command returns an empty collection. This option constrains the search to objects in the clock network only. This option is mutually exclusive with the **-from** option.

-endpoints_only

Includes only the timing endpoints in the result.

-only_cells

Includes only cells in the timing fanout of the *from_list* and not pins or ports.

-flat

There are two major modes in which the **all_fanout** command functions: hierarchical (default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current object are included in the result. In flat mode, the only non-leaf objects in the result are hierarchical from pins.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-step_into_hierarchy

You can use this option only in hierarchical mode with either the **-levels** or **-pin_levels** option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the output pins are considered a single level away from the related input pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related input pins.

-levels *level_count*

The traversal stops when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the object.

-pin_levels *pin_count*

The traversal stops when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the object.

-trace_arcs timing | enabled | all

Specifies the type of combinational arcs to trace during the traversal.

- **timing** (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- **enabled** - Permits the tracing of all enabled arcs and disregards case analysis values.
- **all** - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

-continue_trace *pin_types*

This option permits the user to override the default behavior of the traversal which is to stop at all timing endpoints. When used, the travsal is permitted through endpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- **generated_clock_source** - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

DESCRIPTION

The **all_fanout** command creates a collection of objects in the timing fanout of specified pins/ports or nets in the design. A pin is considered to be in the timing fanout of an object if there is a timing path through combinational logic from that object to the pin (see also the **-trace_arcs** option). The fanout stops at the inputs to registers (sequential cells). The from objects are specified using either **-clock_tree** or **-from** *from_list*.

If a current instance in the design is not the top level of the hierarchy, only objects within the current instance are returned.

The fanout cone can be topologically restricted by using the **-through** and **-to** options.

EXAMPLES

This example shows the timing fanout of a port in the design. The fanout includes the reg3 register.

```
pt_shell> query_objects [all_fanout -from in1]
{"in1", "reg3/D"}
```

This example shows the difference between the hierarchical and flat modes of **all_fanout**. The object in the *from_list* is the H3/z1 output pin of a hierarchical cell, which is connected to the H4/a input pin of another hierarchical cell. H4 contains the U1 leaf cell with input A and output Z. The first command is in hierarchical mode and shows that hierarchical pins are included in the result. The second command is in leaf mode, and leaf pins from the lower level are included.

```
pt_shell> query_objects [all_fanout -from H3/z1]
{"H3/z1", "H4/a", "H4/z", "reg2/D"}

pt_shell> query_objects [all_fanout -from H3/z1 -flat]
{"H3/z1", "H4/U1/A", "H4/U1/Z", "reg2/D"}
```

This example shows how to get the fanout of reg1/CP on paths passing through u1/Z and U3/Z.

```
pt_shell> query_objects [all_fanout -from reg1/CP -through U1/Z \
          -through U3/A]
{"out", "U3/Z", "U3/A", "U1/Z", "U1/A", "reg1/Q", "reg1/CP"}
```

SEE ALSO

```
all_fanin(2)
current_instance(2)
report_transitive_fanin(2)
report_transitive_fanout(2)
```

all_inputs

Creates a collection of all input ports in the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

```
collection all_inputs
   [-level_sensitive]
   [-exclude_clock_ports]
   [-edge_triggered]
   [-clock clock_name]
```

Data Types

```
clock_name      list
```

ARGUMENTS

-level_sensitive

Only considers ports with level-sensitive input delay. This is specified by the **set_input_delay 2 -clock CLK -level_sensitive IN1** command.

-exclude_clock_ports

Excludes input ports which serve as clock sources.

-edge_triggered

Only considers ports with edge-triggered input delay. This is specified by **set_input_delay 2 -clock CLK IN2**.

-clock *clock_name*

Only considers ports with input delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

DESCRIPTION

The **all_inputs** command creates a collection of all input or inout ports in the current design. You can limit the contents of the collection by specifying the type of input delay that must be on a port.

You can remove clock source ports from the resulting collection by specifying *-exclude_clock_ports* option.

If you want only certain ports, use the **get_ports** command to create a collection of ports matching a specific

pattern and optionally passing filter criteria.

When issued from the command prompt, the **all_inputs** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example specifies a driving cell for all input ports.

```
pt_shell> set_driving_cell -lib_cell FFD3 -pin Q [all_inputs]
```

SEE ALSO

```
collections(2)
get_ports(2)
report_port(2)
query_objects(2)
set_driving_cell(2)
set_input_delay(2)
collection_result_display_limit(3)
```

all_instances

Creates a collection of all instances of a specific design or library cell in the current design, relative to the current instance. You can assign the resulting collection of cells to a variable or pass it into another command.

SYNTAX

```
collection all_instances
   [-hierarchy]
   object_spec
```

Data Types

```
object_spec     string
```

ARGUMENTS

-hierarchy

Searches for instances in all levels of instance hierarchy below the current instance. By default, only instances from the current level of hierarchy are considered.

object_spec

Specifies the target design or library cell. This can be a design collection, lib_cell collection, or name.

DESCRIPTION

The **all_instances** command creates a collection of cells that are instances of a design or library cell. The search for instances is made relative to the current instance within the current design. By default, **all_instances** considers only instances at the current level of the hierarchy. If you use the **-hierarchy** option, the search continues throughout the hierarchy.

The *object_spec* can be a simple name. In this case, any instance of a design or lib_cell with that name will match. Alternatively, the *object_spec* can be a collection of exactly one design or one library cell. Any other collection results in an error message. Using the collection can help focus the search for instances of specific designs or lib_cells, especially after **swap_cell** has been used.

When issued from the command prompt, **all_instances** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example uses **all_instances** to get the instances of the design 'low' in the current level of hierarchy.

```
pt_shell> all_instances low
{"U1", "U3"}
```

The following example uses **-hierarchy** to display instances of a design across multiple levels of hierarchy.

```
pt_shell> query_objects [all_instances low -hierarchy]
{"U1", "U2/U1", "U3"}
```

In the following example, there are two instances of design "inter". One of them is swapped for a new design. The difference between using **all_instances** with a name and a collection of one design is shown.

```
pt_shell> swap_cell i1 inter_2.db:inter
 ...unlinking i1
1
pt_shell> all_instances inter
{"i1", "i2"}
pt_shell> all_instances [get_designs inter_2.db:inter]
{"i1"}
```

SEE ALSO

```
collections(2)
current_design(2)
current_instance(2)
get_cells(2)
get_designs(2)
get_lib_cells(2)
list_designs(2)
query_objects(2)
collection_result_display_limit(3)
```

all_outputs

Creates a collection of all output ports in the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

```
collection all_outputs
   [-level_sensitive]
   [-edge_triggered]
   [-clock clock_name]
```

Data Types

```
clock_name     list
```

ARGUMENTS

-level_sensitive

Only considers ports with level-sensitive output delay. This is specified by **set_output_delay 2 -clock CLK -level_sensitive IN1**.

-edge_triggered

Only considers ports with edge-triggered output delay. This is specified by **set_output_delay 2 -clock CLK IN2**.

-clock *clock_name*

Only considers ports with output delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

DESCRIPTION

The **all_outputs** command creates a collection of all output or inout ports in the current design. You can limit the contents of the collection by specifying the type of output delay that must be on a port.

If you want only certain ports, use **get_ports** to create a collection of ports matching a specific pattern and optionally passing filter criteria.

When issued from the command prompt, **all_outputs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example specifies a pin capacitance for all output ports.

```
pt_shell> set_load 4.56 [all_outputs]
```

The following example shows the names all of the output ports with output delay relative to PHI1.

```
pt_shell> all_outputs -clock PHI1
```

SEE ALSO

```
collections(2)
get_ports(2)
report_port(2)
query_objects(2)
set_output_delay(2)
collection_result_display_limit(3)
```

all_registers

Creates a collection of register cells or pins. You can assign the resulting collection to a variable or pass it to another command.

SYNTAX

```
collection all_registers
   [-clock clock_name]
   [-rise_clock rise_clock_name]
   [-fall_clock fall_clock_name]
   [-cells]
   [-data_pins]
   [-clock_pins]
   [-slave_clock_pins]
   [-async_pins]
   [-output_pins]
   [-level_sensitive]
   [-edge_triggered]
   [-master_slave]
   [-no_hierarchy]
```

Data Types

```
clock_name              list
rise_clock_name         list
fall_clock_name         list
```

ARGUMENTS

-clock *clock_name*

Considers all registers clocked by *clock_name*. This is either the name of a clock, or a collection containing a clock. For example, all registers whose clock pins are in the fanout of the specified clock.

-rise_clock *rise_clock_name*

Considers all registers clocked by *rise_clock_name* and having open edge effectively the rising clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops without any inversion on the clock path or falling edge triggered flip-flops with inversion on the clock path.

-fall_clock *fall_clock_name*

Considers all registers clocked by *fall_clock_name* and having open edge effectively falling the clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops with inversion on the clock path or falling edge triggered flip-flops without any inversion on the clock path.

-cells

Collects cells; this is the default behavior of this command. The cells are registers and are further limited by other command options.

-data_pins

Collects register data pins. The collection can be limited by other command options.

-clock_pins

Collects register clock pins. The collection can be limited by other command options.

-slave_clock_pins

Collects register slave clock pins (the slave clock pins of master-slave registers). Specify slave clock pins as **clocked_on_also** in the library.

-async_pins

Collects asynchronous preset or clear pins.

-output_pins

Collects register output pins.

-level_sensitive

Considers only level-sensitive latches.

-edge_triggered

Considers only edge-triggered flip-flops.

-master_slave

Considers only master or slave register cells.

-no_hierarchy

Considers only the current instance; does not descend the hierarchy.

DESCRIPTION

The **all_registers** command creates a collection of pins or cells related to registers.

When issued from the command prompt, **all_registers** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; to change this maximum, set the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

This example performs a trace from a port to all register clock pins which are clocked by CLK.

```
pt_shell> report_timing -from [get_ports {CLK}] \
```

```
          -to [all_registers -clock CLK -clock_pins]

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : counter
 ...
****************************************

  Startpoint: CLK (input port clocked by CLK)
  Endpoint: ffa/CP (internal pin)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  input external delay                     0.00       0.00 r
  CLK (in)                                 0.00       0.00 r
  ffa/CP (FD2)                             0.00       0.00 r
  data arrival time                                   0.00
  -----------------------------------------------------------
  (Path is unconstrained)
```

The following example performs a trace from all registers clock pins, which are triggered by clock rising edge.

```
pt_shell> report_timing -from [all_registers -rise_clock \
            CLK -clock_pins]

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : li_FD3x
 ...
****************************************

  Startpoint: ff3 (falling edge-triggered flip-flop clocked by clk')
  Endpoint: ff4 (falling edge-triggered flip-flop clocked by clk')
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk' (fall edge)                   0.00       0.00
  clock network delay (ideal)              0.00       0.00
  ff3/CP (FD3x)                            0.00       0.00 f
  ff3/Q (FD3x)                             1.44       1.44 f
  iv2/Z (IVA)                              0.31       1.75 r
  ff4/D (FD3x)                             0.00       1.75 r
  data arrival time                                   1.75

  clock clk' (fall edge)                  10.00      10.00
  clock network delay (ideal)              0.00      10.00
  ff4/CP (FD3x)                                      10.00 f
  library setup time                      -0.90       9.10
  data required time                                  9.10
  -----------------------------------------------------------
  data required time                                  9.10
  data arrival time                                  -1.75
  -----------------------------------------------------------
  slack (MET)                                         7.35
```

SEE ALSO

```
collections(2)
get_cells(2)
get_pins(2)
collection_result_display_limit(3)
```

append_to_collection

Adds objects to a collection and modifies a variable.

SYNTAX

```
collection append_to_collection
   [-unique]
   var_name
   object_spec
```

Data Types

var_name	collection
object_spec	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

var_name

Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec

Specifies a list of named objects or collections to add.

DESCRIPTION

The **append_to_collection** command allows you to add elements to a collection. This command treats the variable name given by the *var_name* option as a collection, and it appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from the *object_spec* as its value. If the variable does exist and it does not contain a collection, it is an error.

The result of the command is the collection that was initially referenced by the *var_name* option, or the collection created if the variable did not exist.

The **append_to_collection** command provides the same semantics as the common use of the **add_to_collection** command; however, this command shows significant improvements in performance.

An example of replacing the **add_to_collection** command with the **append_to_collection** command is provided below. For example,

```
set var_name [add_to_collection $var_name $objs]
```

Using the **append_to_collection** command, the command becomes:

```
append_to_collection var_name $objs
```

The **append_to_collection** command can be much more efficient than the **add_to_collection** command if you are building up a collection in a loop. The arguments of the command have the same restrictions as the **add_to_collection** command. For more information about these restrictions, see the **add_to_collection** man page.

EXAMPLES

The following example from PrimeTime shows how a collection can be built up using the **append_to_collection** command:

```
pt_shell> set xports
Error: can't read "xports": no such variable
        Use error_info for more info. (CMD-013)
pt_shell> append_to_collection xports [get_ports in*]
{"in0", "in1", "in2"}
pt_shell> append_to_collection xports CLOCK
{"in0", "in1", "in2", "CLOCK"}
```

SEE ALSO

add_to_collection(2)
foreach_in_collection(2)
index_collection(2)
remove_from_collection(2)
sizeof_collection(2)

apropos

Searches the command database for a pattern.

SYNTAX

```
string apropos
   [-symbols_only]
   pattern
```

Data Types

```
pattern      string
```

ARGUMENTS

-symbols_only

Searches only command and option names.

pattern

Searches for the specified *pattern*.

DESCRIPTION

The **apropos** command searches the command and option database for all commands that contain the specified *pattern*. The *pattern* argument can include the wildcard characters asterisk (*) and question mark (?). The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though **help -verbose** was used with the command.

Whereas **help** looks only at command names, **apropos** looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the **-symbols_only** option.

When searching for dash options, do not include the leading dash. Search only for the name.

EXAMPLES

In the following example, assume that the **get_cells** and **get_designs** commands have the **-exact** option. Note

that without the **-symbols_only** option, the first search picks up commands which have the string "exact" in the one-line description.

```
prompt> apropos exact
 get_cells            # Create a collection of cells
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match cell names against patterns)

 get_designs          # Create a collection of designs
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match design names against patterns)

 real_time            # Return the exact time of day

prompt> apropos exact -symbols_only
 get_cells            # Create a collection of cells
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match cell names against patterns)

 get_designs          # Create a collection of designs
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match design names against patterns)
```

SEE ALSO

help(2)

associate_supply_set

Associates a supply set handle to another supply set handle or supply set reference.

SYNTAX

```
status associate_supply_set
   -handle supply_set_handle
   supply_set_name
```

Data Types

```
supply_set_handle     string
supply_set_name       string
```

ARGUMENTS

-handle *supply_set_handle*

Specifies the supply set handle on which the action is performed.

This is a required argument.

supply_set_name

Specifies the name of the supply set to which the supply set handle is associated. The name can be the name of a supply set or a supply set handle.

This is a required argument.

DESCRIPTION

The **associate_supply_set** command associates the supply set handle specified with the **-handle** option to the specified supply set. After the association, the tool treats corresponding functions in each supply set to be virtually connected. Therefore, the supply sets inherit common switching behavior and must eventually be resolved to the same physical supply net.

A supply set is an abstract collection of supply nets, consisting of two supply functions, power and ground.

EXAMPLES

The following example associates primary supply handle of domain PD1 to the SS1 supply set.

```
prompt> associate_supply_set SS1 -handle PD1.primary
1
```

SEE ALSO

```
create_power_domain(2)
create_supply_set(2)
```

calculate_effective_resistance

Calculates the effective resistance between a given pin and the taps.

SYNTAX

```
int calculate_effective_resistance
   [-pg_pins pg_pin]
   [-cells   cells]
   [-pin_list_file  file_name]
   [-all]
   [-noexpand]
   supply_nets supply_net
   [-output_file_name file_name]
```

Data Types

```
list    pg_pins
list    cells
string  file_name
list    supply_nets
```

ARGUMENTS

-pg_pins *pg_pins*

Specifies targeted pg_pins to be calculated. This option takes both list of names and collections. .

-cells *cell*

Specifies cells of targeted PG pins to be calculated. This option takes both list of names and collections.

-all

Specifies all PG pins to be calculated. This option takes both list of names and collections.

-noexpand

Prevents the listed *supply_nets* from being expanded to include the supply nets that are connected to the listed supply nets by MTCMOS switch cells (also known as power management or PM cells). By default, the list of supply nets is expanded, which has an impact on pin selection. For PG network connectivity, virtual nets are always connected.

-pin_list_file *file_name*

Specifies the file name in which the full hierarchy pin names are listed. This option has the higher priority than the -pg_pins and -cells options.

-output_file_name

Specifies the name of the output file.

supply_net ***list***

Specifies supply_net of targeted PG pins to be calculated. This option filters out unrelated PG pins specified by the -all, -pg_pins and -cells options.

DESCRIPTION

The *calculate_effective_resistance* command calculates the effective resistance between a given pin and the taps. The taps are created with the *create_taps* command.

By default, the number of cores used for effective resistance calculation is 4. To maximize the efficiency, run the **set_host_options** command with the -max_cores option and set the number of core to 16.

EXAMPLES

The following example calculates the effective resistance of all PG pins of VDD net.

```
pr_shell> calculate_effective_resistance -pg_pins [get_pg_pins -hier] VDD
Or
pr_shell> calculate_effective_resistance -all VDD
```

The following example calculates the effective resistance of all PG pins of the current design.

```
pr_shell> calculate_effective_resistance -pg_pins [get_pg_pins] [get_supply_nets]
```

The following example calculates the effective resistance of all VSS PG pins of cells started with "inst".

```
pr_shell> calculate_effective_resistance -cells [get_cells inst* ] VSS
```

The following example calculates the effective resistance of the instA/VDD and instB/VSS pins by the pin_list_file.

```
pr_shell> exec cat pin_list
instA/VDD
instB/VSS
pr_shell> calculate_effective_resistance -pin_list_file pin_list [get_supply_net]
```

SEE ALSO

```
create_taps
set_rail_options
```

calculate_minimum_path_resistances

Calculates the minimum path resistance from each point in the power and ground network to any reachable tap point.

SYNTAX

```
status calculate_minimum_path_resistances
   [-result rail_result_name]
   [-replace]
   [-noexpand]
   supply_nets
```

Data Types

```
string rail_result_name
list supply_nets
```

ARGUMENTS

-result *rail_result_name*

Assigns the name *rail_result_name* to the rail result that is to be created by the command. When not specified, the command assigns a unique name to the result.

-replace

Overwrites an existing rail result with the one generated by the command. When used with the -result option, the specified *rail_result_name* is overwritten with the result generated by the command.

-noexpand

Prevents the listed *supply_nets* from being expanded to include the supply nets that are connected to the listed nets through MTCMOS switch cells (also known as power management cells). By default, the command expands the listed supply nets.

supply_nets

Lists the supply nets to be processed. Each element in the list can be a supply net collection or name pattern. The list of supply nets is automatically expanded to include the supply nets that are connected to those listed through MTCMOS switch cells. To suppress the expansion, use the -noexpand option.

DESCRIPTION

This command calculates the path resistances from each point in the power and ground network to the tap point on any of the target nets that are reachable along the least resistance path. The result of this calculation is available for viewing in the resistivity map. This map is often useful for debugging rail analysis issues.

When the command run is complete, a new rail result is created in the name assigned by the command. To specify a name for the rail result, use the -result option. To overwrite an existing named result with the one created in this command run, use the -replace option.

See the **current_rail_result** man page for more information on rail results.

By default, to analyze all the connected power domains together, the command expands the nets specified in *supply_nets* to include all of the supply nets that are connected to the listed nets through the MTCMOS switch cells (also known as power management cells). To suppress this expansion can be suppressed, use the *-noexpand* option.

Note that supply net parasitics must be read into the session before executing the **calculate_minimum_path_resistances** command.

EXAMPLES

This example calculates the minimum path resistances for the nets VDD1 and VDD2.

```
pr_shell> calculate_minimum_path_resistances {VDD1 VDD2}
```

This example calculates the minimum path resistances for the net VDD and assigns a name to the rail result to be created.

```
pr_shell> calculate_minimum_path_resistances -result MyVDDResult [get_supply_net VDD]
```

SEE ALSO

calculate_rail_voltage(2)
current_rail_result(2)
read_supply_parasitics(2)

calculate_rail_voltage

Performs rail analysis to calculate power and ground pin supply voltages.

SYNTAX

```
status calculate_rail_voltage
   [-static | -dynamic]
   [-start_time time]
   [-end_time time]
   [-step_size time]
   [-total_steps count]
   [-fsdb_file file_name]
   [-result rail_result_name]
   [-replace]
   [-noexpand]
   [-resistivity]
   [-error_cell error_cell_name]
   [-voltage_threshold value]
   [-rush_current_dir dir_name]
   [-force]
   [-include_dynamic_currents]
   [-report_start_offset time_offset]
   supply_nets
```

Data Types

```
float time
int count
string file_name
string rail_result_name
string error_cell_name
float value
string dir_name
list supply_nets
float time_offset
```

ARGUMENTS

-static

Specifies the rail analysis mode to be *static*. This option is incompatible with the -dynamic option.

If neither the -static nor -dynamic analysis mode is specified, the rail analysis mode is taken from the **update_currents** command.

-dynamic

Specifies the rail analysis mode to be *dynamic*. This option is incompatible with the -static option.

If neither the -static nor -dynamic analysis mode is specified, the tool determines the rail analysis mode based on the settings in the **update_currents** command.

-start_time *time*

Specifies the start time for the dynamic analysis, using the time unit defined in the technology library. The start time is the time relative to the reference time (or the VCD start time) reported by the **update_currents** command.

-end_time *time*

Specifies the end time for the dynamic analysis, using the time unit defined in the technology library. The end time is the time relative to the reference time (or the VCD start time) reported by the **update_currents** command.

-step_size *time*

Specifies the simulation time step size for dynamic analysis, using the time unit defined in the technology library.

-total_steps *count*

Specifies the total number of time steps for performing dynamic analysis.

-fsdb_file *file_name*

Specifies an output FSDB file in which current waveforms and voltage drop waveforms are to be saved.

-result *rail_result_name*

Assigns the name *rail_result_name* to the output rail result. When not specified, the command assigns a unique name to the result.

-replace

Overwrites an existing rail result with the one generated by the command. When used with the -result option, the specified *rail_result_name* is overwritten with the one generated by the command.

-noexpand

Prevents the listed *supply_nets* from being expanded to include the supply nets that are connected to the listed nets through MTCMOS switch cells (also known as power management cells). By default, the command expands the listed supply nets.

-resistivity

Calculates minimum path resistance during voltage drop analysis. The result is written into the same rail result with voltage drop and bounce results.

-error_cell *error_cell_name*

Creates a Milkyway error cell named *error_cell_name* which can be viewed in the IC Compiler error browser. If the voltage threshold is not specified using the -voltage_threshold option, the default voltage threshold value is zero.

-voltage_threshold *value*

Specifies the voltage threshold value in the unit of volt for the error cell generation. The specified value is an absolute number for both power and ground nets. When this option is used with the -error_cell option, a Milkyway error cell will be created which can be viewed from the IC Compiler error browser. When this option is used without the -error_cell option, an in-memory error cell will be created which can be viewed from the PrimeRail error browser. All geometries whose voltage drop value is larger than the voltage threshold will be stored to the error cell.

-rush_current_dir *dir_name*

Specifies the name of the directory which contains the FSDB files that are generated by the **calculate_rush_currents** command. When not specified, the command searches for the rush current waveform files in the $rail_database/fsdb directory. This option is valid only for dynamic analysis.

-force

Ignores the maximum limit for simulation steps. When not specified, the command aborts when the required number of simulation steps exceeds the maximum limit which is set to 10,000.

-include_dynamic_currents

Calculates average, peak and RMS currents of each PG network wire during dynamic analysis. These currents are used by the **report_em_violations** command for dynamic electromigration analysis. When not specified, only average, peak and RMS voltages of each PG network node are calculated during dynamic analysis.

-report_start_offset *time_offset*

Specifies a time offset from the beginning of the simulation window. Dynamic analysis starts from the beginning of the simulation window regardless of this option value. However, analysis results are reported and saved only after the specified amount of time has passed. The option value must be specified using the time unit defined in the technology library.

Affected results include minimum, maximum, and average power grid node voltage values and instance-based effective voltage drop values. Corresponding GUI maps are also affected by this option. When the -include_dynamic_currents option is specified, the minimum, maximum, average, and RMS power grid resistor current values used by EM analysis and subsequent EM map display are affected by this option. FSDB waveforms are not affected by this option.

supply_nets

Lists the supply nets that are to be processed. Each element in the list can be a supply net collection or name pattern. The list of supply nets is automatically expanded to include the supply nets that are connected to those listed through MTCMOS switch cells. To suppress the expansion, use the -noexpand option.

DESCRIPTION

The command calculates voltage drop and bounce values at power and ground pins using the current tap points that are specified with the **create_taps** command. These current tap points are used as ideal voltage sources. The command calculates static (that is, averaged) power and ground pin currents when the -static option is specified, or dynamic (that is, time-varying) power and ground pin currents when the -dynamic option is specified.

When the command run is complete, a new rail result is created in the name assigned by the command. To specify the name for the rail result, use the -result option. To overwrite an existing named result with the one created in this command run, use the -replace option.

See the **current_rail_result** man page for more information on rail results.

Two TCL script files will also be created under the directory $rail_database/design_name/rail/RAIL_RESULT_* after the command finishes. One file is retrieve_result.tcl, which allows users to retrieve rail result in a new PrimeRail session. After retrieving the result, users are able to query rail result, and bring up rail maps. The other file is icc2_open_rail_result.tcl which allows user to open a rail result from an ICC2 session, and then query rail results and display rail maps in ICC2. For more details, see the man page of retrieve_result.tcl and icc2_open_rail_result.tcl.

The nets included in *supply_nets* are automatically expanded to include all of the supply nets that are connected to the listed supply nets through MTCMOS switch cells (that is, power management cells). All the connected power domains are analyzed together. This expansion can be suppressed by using the -noexpand option.

Note that before this command is executed, supply net parasitics must be read into the session by using the **read_supply_parasitics** command, and currents should be assigned to power or ground pins by using the **update_currents**, or **set_rail_power**, or **set_rail_current** command.

If in-rush current analysis was performed before this command is executed (whether in the same or a previous session), rush current waveforms generated by the **calculate_rush_currents** command are read in from the FSDB files during the analysis. The impact of the rush currents on the listed supply nets is analyzed during the dynamic voltage drop analysis.

Alternatively, in-rush current analysis can be performed as part of dynamic voltage drop analysis. Use the **set_switched_net_state** command to specify the current state of virtual power domains of the main power domains being analyzed. To report the current state of virtual power domains in the design, use the **report_switched_net_state** command. Each virtual power domain is in ON state unless its state is changed using the **set_switched_net_state** command. See the **set_switched_net_state** command man page for further details.

Once specified, the current state of the virtual power domains is taken into consideration during dynamic voltage drop analysis.

If the current state of a virtual power domain is ON, the virtual power grid is connected to the power grid of the main power domain. In this case, the on-resistance values of the MTCMOS switch cells, which connect the main and virtual power grids, are used.

If the current state is OFF, the power grid of the virtual power domain is disconnected from that of the main power domain. In this case, the total off-state leakage current of the virtual power domain is calculated and distributed to the matching main power domain.

If the current state is ramp_up, in-rush current analysis is performed for the virtual power domain. In-rush current waveforms are calculated for the MTCMOS switch cells which connect the main and virtual power grids. The total in-rush current of the virtual power domain is distributed to the matching main power domain.

When in-rush current analysis is performed as part of dynamic voltage drop analysis, the start and end simulation times are automatically adjusted using the following rules.

Determination of the simulation start time:

1. If the start time is specified by the user, use this value.
2. Otherwise, use the value whichever is smaller between earliest start time of the switching current waveforms from **update_currents** earliest ramp_up event time for the ramp_up virtual supply nets

Determination of the simulation end time:

1. If the end time is specified by the user, use this value.
2. Otherwise, use the value whichever is larger between last end time of the switching current waveforms from **update_currents** last wake-up time for the ramp_up virtual supply nets

If one or more windows are specified as **-time** option value for **read_vcd** command, these windows are automatically detected and dynamic rail analysis is performed on each of these windows. If there is a time gap between these windows, analysis is not performed for that interval to minimize the overall analysis time. Therefore, a rail result produced from multiple window simulation contains power grid node voltage (maximum, minimum, and average) and resistor current (maximum, average, and RMS) values which are calculated over the given windows. Same is true for effective instance voltage drop values (maximum, minimum, and average). Please note multiple simulation windows are supported only for dynamic analysis with an ideal, lumped, or SPICE package model. **-start_time** and **-end_time** options are ignored when multiple simulation windows are present.

EXAMPLES

The following example creates tap points from top-level power and ground pins, reads in the supply net parasitics, and performs voltage drop analysis. Run the **report_rail_result** command to generate a verbose report on supply nets and a snapshots of taps in the result.

```
pr_shell> create_taps -top_pg
pr_shell> read_supply_parasitics VDD
pr_shell> calculate_rail_voltage VDD
pr_shell> report_rail_results -verbose

************************************
RAIL_RESULT has rail result
  Supply nets in the rail result: VDD

  Taps in the result: TAP_1 TAP_2
```

The following example performs voltage drop analysis and resistivity calculation at the same time, and saves the result in the name ir_resistivity:

```
pr_shell> calculate_rail_voltage VDD -resistivity -result ir_resistivity
```

Assume VDD is connected to virtual nets VDD1 and VDD2 through MTCMOS switch cells. The following example first performs voltage drop analysis for VDD, VDD1 and VDD2, and saves the result in the name VDD_all. Then it performs voltage drop analysis for VDD only with the -noexpand option and saves the result in the name VDD_only.

```
pr_shell> calculate_rail_voltage VDD -result VDD_all
pr_shell> calculate_rail_voltage VDD -noexpand -result VDD_only
```

The following example performs static voltage drop analysis with user-defined power without running the **update_currents** command.

```
pr_shell> set rail_power 0.01 inv100/VDD
pr_shell> calculate_rail_voltage VDD -static -result user_assigned_power
```

The following example performs combined in-rush and dynamic voltage drop analysis for the supply net VDD and its virtual supply nets.

```
pr_shell> calculate_rush_currents -start_voltage 0.0 -stop_voltage 1.0 VDD
pr_shell> calculate_rail_voltage VDD -dynamic
```

The following example performs in-rush current analysis as part of dynamic voltage drop analysis for the supply net VDD and its virtual supply net VDDV.

```
pr_shell> set_switched_net_state -state ramp_up VDDV
pr_shell> calculate_rail_voltage VDD -dynamic
```

The following example shows how dynamic rail analysis is performed on the two windows: 100 nsec to 150 ns, and 500 ns to 600 ns.

```
pr_shell> read_vcd dump.vcd -time {100 150 500 600}
pr_shell> calculate_rail_voltage -dynamic {VDD VSS}
```

SEE ALSO

```
calculate_rush_currents(2)
create_taps(2)
current_rail_result(2)
get_taps(2)
read_supply_parasitics(2)
read_vcd(2)
report_em_violations(2)
remove_rail_result(2)
rename_rail_result(2)
report_rail_results(2)
report_switched_net_state(2)
report_taps(2)
set_rail_current(2)
set_rail_power(2)
scale_rail_power(2)
set_switched_net_state(2)
update_currents(2)
```

calculate_rush_currents

Performs inrush current analysis for virtual supply nets.

SYNTAX

```
status calculate_rush_currents
   [-start_voltage start_voltage]
   [-stop_voltage stop_voltage]
   [-threshold threshold_value]
   [-fsdb_dir dir_name]
   [-noexpand]
   [-include_main_net_ir_drop]
   [-force]
   [-reuse]
   supply_nets
```

Data Types

```
float start_voltage
float stop_voltage
float threshold_value
string dir_name
list supply_nets
```

ARGUMENTS

-start_voltage *start_voltage*

Specifies the voltage of the virtual supply net when it is in sleep mode. When not specified, a value of 0.0V is applied for a virtual power net. For a virtual ground net, the operating voltage of the related power supply net is used instead.

-stop_voltage *stop_voltage*

Specifies the voltage of the virtual supply net when it is fully charged. When not specified, the operating voltage is used for a virtual power net, and a value of 0.0V is used for a virtual ground net. The wake-up time of the supply net is the time when its voltage reaches *stop_voltage*.

-threshold *threshold_value*

Specifies how the *stop_voltage* of a virtual supply net is calculated if the -stop_voltage option is not specified. For a virtual power net, the *stop_voltage* is calculated by multiplying (1.0-threshold_value) to its operating voltage. For a virtual ground net, the *stop_voltage* is calculated by multiplying *threshold_value* to the related power supply voltage. When both the -stop_voltage and -threshold options are specified, the -threshold option is ignored.

The default *threshold_value* is 0.01, which is used when neither the -stop_voltage nor -threshold option is

specified.

-fsdb_dir *dir_name*

Specifies the directory name where FSDB files are to be saved. By default, the files are saved to the $rail_database/fsdb directory.

-noexpand

Avoids expanding the listed *supply_nets* to include the virtual supply nets that are connected by multithreshold-CMOS switch cells. The multithreshold-CMOS cells are also referred as power management (PM) cells. By default, the command expands the list of supply nets.

-include_main_net_ir_drop

Includes main net voltage drop during inrush current analysis. By default, main net IR drop is ignored.

-force

Ignores the maximum limit for simulation steps. When not specified, the command aborts if the required number of simulation steps exceeds the maximum limit, which is set to 10,000.

-reuse

Uses the stored switching-cell power-up sequence from the previous session. When selected, users do not have to run **update_currents** or **create_mtcmos_switch_cell_sequence** before running rush current analysis with this option.

supply_nets

Lists the top-level or virtual supply nets. Each element of the list can be a supply net collection or a name pattern. By default, the list of supply nets is expanded to include the virtual supply nets that are connected through multithreshold switch cells. Inrush current analysis is then performed for each virtual supply net found during the expansion.

To avoid the expansion, use the -noexpand option. If this is the case, only virtual supply nets must be specified.

DESCRIPTION

The command performs inrush current analysis for each virtual supply net which is either derived from or specified in *supply_nets*. The nets included in *supply_nets* are automatically expanded to include all of their virtual nets connected through the multithreshold switch cells (that is, power management cells). Avoid this expansion by using the -noexpand option.

Each virtual supply net is independently analyzed based on the power-up sequence which is either automatically derived from the VCD file or manually specified by using the **create_mtcmos_switch_cell_sequence** command. When VCD-based power-up sequence is used, the transition time for each switching pin is determined by the signal pin transition time derived from timing analysis.

When user-specified power-up sequence is used, the transition time for each switching pin is defined by using the **create_mtcmos_switch_cell_sequence** command. If power-up sequence is specified but no transition time is defined for switching pins, zero transition time is applied.

The total off-state leakage current for each virtual supply net is also calculated during inrush current analysis, based on IV curves of the PM cells. Average leakage power consumption of each cell instance connected to the virtual supply net is also available.

Once the **calculate_rush_currents** command is completed, the following statistics are reported for each virtual supply net:

```
Total off-state leakage current
Inrush current analysis simulation period
Total wire capacitance
Maximum total intrinsic capacitance
Peak rush current
Time when peak rush current occurred
Time when stop voltage was reached
Power-up duration
Partial-on duration (time between stop voltage was reached and last event time)
Time when the virtual supply net became fully on (same as last event time)
```

Wake-up voltage waveforms and total rush current waveforms for each virtual net are saved in the *virtual_net_name*.fsdb file under the FSDB directory. Each FSDB file contains rush current waveforms at the main PG pins of each PM cell that is connected to the virtual supply net. View these voltage and current waveforms with a waveform viewer that supports the FSDB format, such as nWave.

To obtain accurate inrush current analysis results, it is important to use properly characterized Synopsys Liberty libraries which contain voltage- and state-dependent intrinsic RC values at each standard cell PG pin and correct IV curves for each PM cell master.

EXAMPLES

The following example performs inrush current analysis for the virtual supply net VDDV of the top-level supply net VDD which has only one virtual net. The virtual net voltage in sleep mode is 0.0V and the wake-up time is reached when the virtual net voltage reaches 1.0V. The impact of main net voltage drops is also considered during inrush current analysis.

```
pr_shell> calculate_rush_currents -start_voltage 0.0 -stop_voltage 1.0 \\
         -include_main_net_ir_drop VDD

Information: Expanding operation to include supply net VDDV connected to the specified supply net VDI
Information: VDD is not a virtual supply net -
        Skipped in-rush current analysis.
Starting in-rush current analysis for VDDV
Information: In-rush current analysis with main net IR drop
Minimum path resistances were calculated and annotated to main PG pins of PM cells.
Minimum path resistance range from VDD (main net) to VDDV: 2.594e+00 to 1.044e+01 Ohms
virtual power domain: VDDV
  supply voltage: 1.100E+00
  start voltage : 0.000E+00
  stop voltage  : 1.000E+00
  number of virtual PG pins: 490
  number of STD PG pins: 13401

VDDV: Off-state leakage current calculation stats
  Total off-state leakage current: 9.185E-02 [uA]
  Off-state virtual net voltage  : 8.825E+00 [mV]
  Total cell leakage current     : 1.145E+01 [uA]
  Ratio between off-state and total cell leakage currents: 8.022E-03

Source of PM cell events: VCD
  First event time: 5.000E+00 [nsec]
  Last event time : 2.853E+01 [nsec]
VDDV: In-rush current analysis stats
  In-rush analysis simulation period: 5.000E+00 to 2.854E+01 [nsec]
```

```
    Total wire capacitance            : 2.740E+00 [pF]
    Max total intrinsic capacitance   : 3.289E+01 [pF]
    Peak rush current                 : 1.860E+01 [mA]
    Peak rush current at time         : 8.420E+00 [nsec]
    Stop voltage reached at time      : 9.880E+00 [nsec]
    Power-up duration                 : 4.880E+00 [nsec]
    Partial-on duration               : 1.865E+01 [nsec]
    Full-on at time                   : 2.853E+01 [nsec]
  Information: rush current waveforms were written to /slowfs/dept5372w/test_design/RAIL_DATABASE/fsdb/
  1
```

The following example performs inrush current analysis only for the virtual supply net VDDV1 of the top-level supply net VDD which has three virtual supply nets, VDDV1, VDDV2, and VDDV3.

```
pr_shell> calculate_rush_currents -start_voltage 0.0 -stop_voltage 1.0 -noexpand VDDV1
```

The following example performs inrush current analysis for the virtual supply nets VDDV1 and VDDV2 of the top-level supply net VDD which has three virtual supply nets VDDV1, VDDV2, and VDDV3. Inrush current analysis is independently performed for VDDV1 and VDDV2.

```
pr_shell> calculate_rush_currents -start_voltage 0.0 -stop_voltage 1.0 -noexpand {VDDV1 VDDV2}
```

SEE ALSO

```
create_mtcmos_switch_cell_sequence(2)
report_mtcmos_switch_cell_sequence(2)
remove_mtcmos_switch_cell_sequence(2)
```

cell_of

Creates a collection of cells of the given pins. The **cell_of** command is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
string cell_of
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Creates a list of pins for which cells to get.

DESCRIPTION

The **cell_of** command creates a collection of cells owned by a list of pins. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the **cell_of** command can be found in the Design Compiler documentation. The supported method for getting pins of cells is to use the **get_cells** command with the *-of_objects* option.

SEE ALSO

get_cells(2)

change_macro_view

Changes the view of the macro that is used.

SYNTAX

```
status change_macro_view
   -reference cell_reference_name
   -view view_name
   [-quiet]
```

Data Types

```
cell_reference_name     string
view_name               string
```

ARGUMENTS

-reference *cell_reference_name*

Specifies the cell reference name for which to change the reference view.

All instances of the cell reference will use the view specified by the -view option with the *view_name* value.

-view *view_name*

Specifies the view name to be used.

.prod icc psyn

Valid values for *view_name* are FRAM and CEL. .prod all

.prod syn

Valid values *view_name* are FRAM and ILM. .prod all

-quiet

Suppresses the messages.

DESCRIPTION

The **change_macro_view** command changes the macro's view. All instances of the cell reference are affected by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

.prod icc psyn

The following example replaces the existing CEL view with the FRAM view for the mem_ctrl macro: .prod all

.prod syn

The following example replaces the existing ILM view with the FRAM view for the mem_ctrl icc-ilm: .prod all

```
prompt> change_macro_view -reference mem_ctrl -view FRAM
```

.prod syn

Limitations

The command is only available in Design Compiler topographical mode. .prod all

.prod icc psyn

The following example specifies that the CEL view is to be used for the mem256x16 reference cell:

```
prompt> change_macro_view -reference mem256x16 -view CEL
```

.prod all

SEE ALSO

```
create_rail_macro_model(2)
create_conn_view(2)
```

change_net_type

Changes the type of a net from signal to power or ground. All the pins connected to this net will be treated as PG pins, not signal pins.

SYNTAX

```
status change_net_type
   -from net_type
   -to net_type
   net_name
```

Data Types

```
string net_type
string net_name
```

ARGUMENTS

net_name

Specifies the name of the net whose net type is to be changed from signal to power or ground.

-from *net_type*

Specifies the net type to be changed. The supported net type is signal.

-to *net_type*

Specifies the net type to change to. The supported net types are power and ground. You can specify only one at a time.

DESCRIPTION

The command changes the internal representation of a net from signal type to PG type. After the net type change, this net can be analyzed for current and voltage violations in PrimeRail.

This command must be run before the **open_mw_cel** command.

EXAMPLES

The following example changes the net type of the signal net analog_power_net to net type power.

```
pr_shell> change_net_type -from signal -to power analog_power_net
1
pr_shell> open_mw_cel -library design my_design
1
pr_shell> get_supply_nets
{VDD VSS fi03_D}
```

SEE ALSO

open_mw_cel(2)

change_pin_type

Changes the type of the signal pin from signal to power or ground.

SYNTAX

```
status change_pin_type
   -to pin_type
   -pin pin_name
   cell_master_name
```

Data Types

```
string pin_type
string pin_name
string cell_master_name
```

ARGUMENTS

cell_master_name

Specifies the name of the cell master which contains the signal pin whose pin type is to be changed.

-pin *pin_name*

Specifies the name of the signal pin whose type is to be changed to power or ground.

-to *pin_type*

Specifies the pin type to change to. The supported pin types are power and ground. You can specify only one at a time.

DESCRIPTION

The command changes the internal representation of a pin from signal type to power or ground type. After the changes, users can use the **get_pg_pin** command to get a collection of the PG pins, and assign current or power to these PG pins.

This command must be issued before the **open_mw_cel** command.

If the command is issued multiple times for the same cell master and the samepin, the last setting overwrites all the previous ones.

EXAMPLES

The following example changes the pin type of the AVDD pin in the cell master ana01 to type power:

```
pr_shell> change_pin_type -to power -pin AVDD ana01
1
pr_shell> open_mw_cel -library design top
1
pr_shell> get_pg_pins *
{... ana01_ins1/AVDD ...}
```

SEE ALSO

open_mw_cel(2)

change_selection

Changes the selection in the GUI.

SYNTAX

```
int change_selection
   [-name slct_bus]
   [-replace]
   [-add]
   [-remove]
   [-toggle]
   [-type object_type]
   [-clock_trees clock_tree_list]
   [collection]
```

Data Types

```
slct_bus          string
object_type       string
clock_tree_list   list
collection        list
```

ARGUMENTS

-name *slct_bus*

Specifies the selection bus to which the change is made. The default is *global*, the default global selection bus.

-replace

Replaces current selection with the objects in the collection. This is the default behavior if no optional parameter is specified.

-add

Adds the objects in the collection to the current selection.

-remove

Removes the objects that are specified in the collection from current selection.

-toggle

Toggles the selection state of the objects that are specified in the collection.

-type *object_type*

Specifies the type to change. Only those items from the collection that are of the type specified are used to

change the selection. If not specified, the entire collection is used. The *object_type* option can be one of the following types:

- design
- port
- net
- cell
- pin
- path (timing path)

-clock_trees *clock_tree_list*

Lists the clock trees available to use to change the selection. The type of change that is applied to the current selection with the clock tree list is specified by the optional arguments of this command.

collection

Specifies the collection of objects to use to change the selection. The type of change that is applied to the current selection with the collection is specified by the optional arguments of this command.

DESCRIPTION

The **change_selection** command changes the selection in the GUI. When selections are changed, the GUI updates all the relevant windows to reflect this change.

A collection of objects and the type of change are given as inputs to the command. The collection of objects could be returned as the result of another command such as the **get_designs** command. If the collection is empty and the **-replace** option is specified (or no option specified), the current selection is cleared.

If the **-type** option is used, only the type of objects specified are used to change the current selection. For example, if a **-type** design is used, only the design objects in the collection are used to change the current selection. If no **-type** option is specified, all the objects in the collection are used to change the current selection. For example, if the **-add** option is used with no **-type**, all the objects regardless of type of object are added to the current selection.

For information about collections, see the **collections** man page.

EXAMPLES

The following example replaces the current selection with the collection of design objects, regardless of type:

```
pt_shell> change_selection [get_designs]
```

This example adds cell U5 to the selection:

```
pt_shell> change_selection -add [get_cells U5]
```

This example removes all the objects of type pin from the current selection:

```
pt_shell> change_selection  -remove -type pin [get_selection]
```

This example clears all the selection:

```
pt_shell> change_selection ""
```

SEE ALSO

```
collections(2)
filter(2)
filter_collection(2)
get_selection(2)
query_objects(2)
```

characterize_context

Captures the timing context of a list of instances.

SYNTAX

```
string characterize_context
   -block block_name
   [-instances cell_list]
   [-name mim_group_name]
```

Data Types

```
block_name      string
cell_list       list
mim_group_name  string
```

ARGUMENTS

-block *block_name*

Specifies the name of the design subblock for which this configuration is to be applied. This option is required.

-instances *cell_list*

Specifies a list of one or more instance names in the current design. These named instances must have the same reference block which is specified by the option **-block** in the same configuration command.

-name *mim_group_name*

Specifies the name of the instance group subset. The instances specified in **-instances** will have their context merged together for a single common block level analysis. This option is required if **-instances** is used, unless the given instances have cover all the instances of the given block.

DESCRIPTION

Captures the timing context of instances of subdesigns from the chip-level timing environment. The timing context of an instance is defined as waveforms of the clocks affecting this instance, input arrival times, output required times, timing exceptions, design rules, propagated constants, input drives, and capacitive loads.

EXAMPLES

The following command derives context information for instance I1 and I2.

```
pt_shell> characterize_context -instances {I1 I2} -block B
```

SEE ALSO

remove_context(2)
report_context(2)
write_context(2)

check_ccs_power

Checks the library for out-of-bounds parasitics, missing current waveforms, erroneous channel resistance ratio, out-of-range leakage currents, high intrinsic capacitances, etc.

SYNTAX

```
int check_ccs_power
   [ -lib lib_name ]
   [ -lib_cell lib_cell_name ]

   string  lib_name
   string  lib_cell_name
```

ARGUMENTS

-lib *lib_name*

Specifies the Milkyway library which needs to be checked.

-lib_cell *lib_cell_name*

Specifies the library cells for which the reporting should be done.

DESCRIPTION

The **check_ccs_power** command reports all the cells which have out-of-bounds parasitics, missing current waveforms for any of the PG Pins, out-of-range leakage currents, high intrinsic capacitances, etc.

EXAMPLES

The following example checks all the linked libraries.

```
pr_shell> check_ccs_power
```

The following example checks the library named "*typical_library*".

```
pr_shell> check_ccs_power -lib typical_library
```

The following example checks the library cell "*typical_library/AND01*"

```
pr_shell> check_ccs_power -lib_cell typical_library/AND01
```

check_constraints

Checks constraints of the current design and reports potential problems.

SYNTAX

```
int check_constraints
   [-verbose]
```

ARGUMENTS

-verbose

Print verbose report of constraint errors.

DESCRIPTION

Invokes PrimeTime GCA to check constraints of the current design for potential problems.

EXAMPLES

The following example shows a constraint checking report:

```
pt_shell> check_constraints -verbose

****************************************
Report : check_constraints
 ...
****************************************

Scenario Violations  Count Waived Description
-------------------------------------------------------------------------------
Design: counter
<Global Violations>     1    0   Scenario independent violations
  error                 1    0
    UNT_0002            1    0   Library 'library' has incomplete units defined.
      1 of 1                 0   Library '/u/zinmgr/test_data/ext_libs/test10k.db:test10k' has
                                    incomplete units defined.
default                 8    0   Default scenario violations
  warning               8    0
    EXD_0001            4    0   Input/inout port 'port' has no input delay specified
```

```
        1 of 4                 0   Input/inout port 'A' has no input delay specified
        2 of 4                 0   Input/inout port 'B' has no input delay specified
        3 of 4                 0   Input/inout port 'C' has no input delay specified
        4 of 4                 0   Input/inout port 'D' has no input delay specified
      EXD_0003            4    0   Output/inout port 'port' has no clock-related output delay
                                      specified
        1 of 4                 0   Output/inout port 'QA' has no clock-related output delay specified
        2 of 4                 0   Output/inout port 'QB' has no clock-related output delay specified
        3 of 4                 0   Output/inout port 'QC' has no clock-related output delay specified
        4 of 4                 0   Output/inout port 'QD' has no clock-related output delay specified
    ------------------------------------------------------------------------------
    Total Error Messages    1    0
    Total Warning Messages  8    0
    Total Info Messages     0    0
```

SEE ALSO

```
check_timing(2)
report_case_analysis(2)
```

check_level_shifters

Alias for the **check_timing -override_defaults {signal level}** command.

SYNTAX

```
string check_level_shifters
   [-verbose]
```

ARGUMENTS

-verbose

Prints a detailed report of all possible violations.

DESCRIPTION

In PrimeTime, the **check_level_shifters** command is an alias for:

```
check_timing -override_defaults {signal_level}
```

SEE ALSO

check_timing(2)

check_mw_lib

Check Milkyway library.

SYNTAX

```
status_value check_mw_lib
   [mw_lib | -lib_id lib_id]
   [-technology]
   [-treat_blockage_as_thin_wire]
```

ARGUMENTS

mw_lib

Specifies the Milkyway library to be checked. This argument and **-lib_id** are mutually exclusive. If both are omitted, the current Milkyway library is used.

-lib_id *lib_id*

Specifies the ID of the Milkyway library to be checked. This argument and *mw_lib* are mutually exclusive. If both are omitted, the current Milkyway library is used.

-technology

Indicates to check that the technology file is loaded and valid.

-treat_blockage_as_thin_wire

Specifies to treat all blockages as thin wire instead of classifying them according to their widths.

DESCRIPTION

Check various information about a Milkyway library.

A status indicating success or failure is returned.

EXAMPLES

The following example checks technology of the current Milkyway library.

```
prompt> check_mw_lib -technology
1
```

SEE ALSO

close_mw_lib(2)
open_mw_lib(2)

check_noise

Checks that the necessary data is available to run the **update_noise** command.

SYNTAX

```
int check_noise
   [-verbose]
   [-beyond_rail]
   [-nosplit]
   [-include noise_driver | noise_immunity]
```

ARGUMENTS

-verbose

Enables verbose mode showing detailed information and pin names; shows cell and pin names that have no model or invalid models.

-beyond_rail

Enables beyond_rail noise analysis check. By default, **check_noise** performs only within_rail analysis.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output, because most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-include noise_driver | noise_immunity

Specifies one of the following types of checking: **noise_driver** and **noise_immunity** (the default).

DESCRIPTION

It is possible to have invalid noise models in a library and run noise analysis without detecting them. This might result in inaccurate results. If the design is large, and it takes a long time to finish the noise analysis, it also causes longer turnaround time. Moreover, it is very important that all pins in the design have correct noise immunity information. Otherwise, when the noise analysis is over, violations are not reported even if noise bumps are large enough to create violations.

To validate the correctness of a design with respect to the noise analysis, run the **check_noise** command before performing noise analysis. It checks any invalid noise model, any pin without a model from the library, and the design. It checks if all pins in the design are 'noise constrained', for example, so their noise immunity can be

calculated.

The command might produce different results before and after **update_timing**. For example, case analysis can disable noise models, and pins can be unconstrained after **update_timing**. It is recommended that you run the **check_noise** command after **update_timing** to consider design context.

EXAMPLES

The following example shows the noise immunity check.

```
pt_shell> check_noise

  Noise immunity type            above_low  below_high
  -------------------------------------------------------
  user hyperbolic curve                  8           8
  user margin                            1           1
  library immunity table                 0           0
  library hyperbolic curve               0           0
  library CCS noise immunity          3245        3245
  library DC noise margin                0           0
  none                                   0           0
```

The following example shows the noise driver check.

```
pt_shell> check_noise -include "noise_driver"

Noise driver check:

  Noise driver type              above_low    below_high
  -------------------------------------------------------
  CCS noise                            520         520
  library IV curve                     121         121
  library resistance                     0           0
  equivalent library pin                 1           1
  user resistance                        1           1
  none                                   1           1
```

The verbose mode shows which pin is not constrained.

```
pt_shell> check_noise -include "noise_driver" -verbose

Noise driver check:

  library pin name                           Region           Status
  ---------------------------------------------------------------------------
  top/inv1/Z                                 above_low        None

  Noise driver type              above_low    below_high
  -------------------------------------------------------
  CCS noise                            520         520
  library IV curve                     121         121
  library resistance                     0           0
  equivalent library pin                 1           1
  user resistance                        1           1
  none                                   1           1
```

SEE ALSO

```
report_noise(2)
set_noise_parameters(2)
update_noise(2)
```

check_power

Shows possible power problems for design.

SYNTAX

```
string check_power
   [-verbose]
   [-significant_digits digits]
   [-override_defaults check_list]
   [-include check_list]
   [-exclude check_list]
```

Data Types

```
digits        integer
check_list    list
```

ARGUMENTS

-verbose

Shows detailed information about potential problems.

-significant_digits *digits*

Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default.

-override_defaults *check_list*

Overrides the checks in **power_check_defaults** variable with those listed in *check_list*. The allowed checks are:

```
no_arrival_window
no_base_clock
out_of_table_range
missing_table
missing_function
```

-include *check_list*

Adds the checks listed in *check_list* to the checks in **power_check_defaults**.

-exclude *check_list*

Subtracts the checks listed in *check_list* from the checks in **power_check_defaults**.

DESCRIPTION

The check_power command checks structure of the design for potential power violations. This command is used to identify possible power calculation problems before updating power or before generating power reports.

This command also prints which checks it performs. If a check reveals a violation, then the command also prints a message about the violation. By default, the message contains a summary of the violation. To get more information about violations, use the **-verbose** option.

This command without any options performs the checks defined by the **power_check_defaults** variable. To change the value of this variable, either redefine it or use the **-override_defaults** option. If the **-override_defaults** option is not used, the final list of checks to be performed is the checks in **power_check_defaults** plus the list of checks given by the **-include** option minus the list of checks given by the **-exclude** option. If the **-override_defaults** option is used with a check_list, the final list of checks to be performed is the checks in the check_list given by the option plus the list of checks given by the **-include** option minus the list of checks given by the **-exclude** option.

The alphabetically ordered list below shows the meaning of each check:

missing_function

Checks if library cells have function statements. The function statements are used during toggle propagation and event simulation.

missing_table

Checks if library cells have leakage power tables and dynamic power tables.

no_arrival_window

Checks whether pins have arrival window or not. This check helps to generate accurate cycle-based average waveform, zero delay VCD power analysis, and RTL VCD power analysis. In order to save the arrival windows user should set the variable timing_save_pin_arrival_and_slack to true before update_timing/update_power. Users seeing this violation are directed to check if timing is correct by using the **check_timing** command.

no_base_clock

Checks whether nets have power base clock or not. During updating power, the fastest clock is used as power base clock for these nets that do not have power base clock. However, this exposes some potential accuracy problem. Users seeing this violation are directed to check if timing is correct by using the **check_timing** command.

out_of_table_range

Checks if there are out-of-range ramp or load violations when looking up power tables for each cells. The out of range values together with cell name, pin name, and library cell name and its range is displayed if the "-verbose" option is used.

SEE ALSO

```
power_check_defaults(3)
check_timing(2)
report_constraint(2)
```

check_supply_net_integrity

Analyzes the routing geometries associated with a supply net to identify common routing problems for reducing the effectiveness of the supply network.

SYNTAX

```
status check_supply_net_integrity
   supply_net
   [-error_view prefix_name]

   [-non_ideal_sourced_floating_shapes]
   [-pg_pin_floating_connection_check disable | minimal | maximal
     [-ignore_touching_floating_pin_shape_errors]
     [-ignore_pins_by_cell_type mask_layout_type_list]
     [-ignore_pins_by_cel cell_list]]
   [-ideal_sourced_dangling_vias]
   [-dangling_pin_shape_edge_check
     [-ignore_pin_shape_layers routing_layer_list]]

   [-ignore_eeq_for_physical_connectivity_tracing
     | -override_eeq_class_to_interpret_as_same_class]
   [-use_conn_view
     [-ignore_conn_view_eeq_for_physical_connectivity_tracing
       | -override_conn_view_eeq_class_to_interpret_as_same_class]]
   [-dont_group_contiguous_floating_shapes]

   [-missing_via_rule {layer_name1 route_type1 layer_name2 route_type2}
     [-missing_via_net_shape_direction {perpendicular (default) | parallel | both}]
     [-max_via_distance_along_parallel_wires distance]
     [-honor_drc_rules]
     [-ignore_minimum_via_criterion]
     [-via_existence_check_partially_enclosed]
     [-via_existence_check_merge_via_cut]
     [-no_stack]
     [-existing_stacked_via_check_mode via_cut | via_enclosure]
     [-allow_stacked_endpoint_vias_outside_of_overlapping_area]
     [-merge_same_layer_shapes {<pin_shape | net_shape | terminal | via_metal_enclosure> ... }]
     [-cut_merged_shape]
     [-ignore_via_enclosure_shape]
     [-ignore_routing_blockage_layers {blockage_layers}]
     [-ignore_routing_blockages blockages]
     [-ignore_other_pg_net_shapes]
     [-ignore_small_overlapping_area area]
     [-filter_layer_net_shapes {{{layer_name [direction]} {[filter_expression]}} ... }]
     [-report_specific_missing_via_in_stacked_via <true | false>]]

   [-enable_discontinuous_connection_check
     [-discontinuous_connection_threshold metal_layer_gap_distance_pairs]
     [-ignore_discontinuous_connection_routing_layers routing_layer_list]
     [-discontinuous_connection_directionality metal_layer_directionality_pairs]
     [-ignore_discontinuous_connection_object_types object_type_spec_list]
     [-ignore_discontinuous_connection_shape_types shape_type_list]
     [-ignore_other_shape_net_types shape_net_type_list]]
```

```
[-short_circuit_check]

[-area_edge_condition <touching (default) | within | intersecting>]
[-included_area_polygon_list {x1 y1 ... xN yN x1 y1} ... }]
[-excluded_area_polygon_list {x1 y1 ... xN yN x1 y1} ... }]
[-result integrity_result_name]
[-replace]
```

Data Types

```
collection supply_net
string layer_name1
string route_type1
string layer_name2
string route_type2
real x1
real y1
real xN
real yN
string prefix_name
string integrity_result_name
float distance
list of strings metal_layer_gap_distance_pairs
list of strings routing_layer_list
list of strings metal_layer_directionality_pairs
list of strings object_type_spec_list
list of strings shape_type_list
list of strings shape_net_type_list
```

ARGUMENTS

supply_net

Specifies the supply nets to be analyzed.

-ideal_sourced_dangling_vias

Checks for all dangling vias that have a physical connection missing on either layer adjacent to the cut layer, but are contiguous with a tap constraint. Note that floating dangling vias are not checked.

This check is grouped with the floating checks because it determines only dangling vias that are not floating. This means that other dangling vias could exist in the design but initially are flagged as an error only with the -non_ideal_sourced_floating_shapes option. To find other dangling vias, fix the design so they are no longer floating and rerun the dangling via check.

-dangling_pin_shape_edge_check

Checks for pin shapes that have pin access edges but no physical connection to the ideal voltage source. The engine does not flag an error on an access edge if it is floating. Use the -pg_pin_floating_connection_check option to check for this particular situation.

-ignore_pin_shape_layers *routing_layer_list*

Skips dangling pin shape checking and floating pin shape connection checking on the specified Milkyway routing layers. The *routing_layer_list* is a list of Milkyway routing layers.

The option is valid only when used with the -pg_pin_floating_connection_check or -

dangling_pin_shape_edge_check option.

-non_ideal_sourced_floating_shapes

Checks for floating supply net segment shapes. Floating net segment shapes are those that have no path to a tap point (that is, an ideal voltage source).

-pg_pin_floating_connection_check disable | minimal | maximal

Checks for PG pins that do not have physical connections to a tap point (that is, an ideal voltage source).

Valid values for this option are

- **disable** (the default)
  Disables the PG pin floating connection check.
- **minimal**
  Flags a pin as nonfloating when at least one pin shape of the pin that has the same electrically equivalent (EEQ) class is not floating. The EEQ classes are determined from the **eeq_class** attributes in the FRAM view or the implied **conn_view_eeq** attribute in the CONN view.
- **maximal**
  Flags a pin as nonfloating when all pin shapes of the pin that are of the same EEQ class are not floating.

  Note that the command traces between pin shapes of the same *eeq_class* or *conn_view_eeq_class* attribute value. When one pin shape of a given value is not floating, all of the other pin shapes with the same value are also not floating. When the maximal setting is set, use the -ignore_eeq_for_physical_connectivity_tracing or -ignore_conn_view_eeq_for_physical_connectivity_tracing option to prevent tracing between pin shapes of the same EEQ value within a cell instance.

  By default, the command checks cell instances of all cell types. To restrict which types of cells to be checked, use the -ignore_pins_by_cell_type option.

-ignore_touching_floating_pin_shape_errors

Allows a floating pin shape if it connects to a given supply net which also touches or overlaps another floating pin shape. The option is valid only when used with the -pg_pin_floating_connection_check option.

-use_conn_view

Derives EEQ class from the cell master's CONN view.

-override_eeq_class_to_interpret_as_same_class

Assumes all pin shapes of a given pin on all cells are of the same eeq_class, irrespective of the *eeq_class* attribute on a pin shape.

-ignore_eeq_for_physical_connectivity_tracing

Prevents physical connectivity tracing from traversing from one EEQ pin shape of the same *eeq_class* to another EEQ pin shape of the same class.

-override_conn_view_eeq_class_to_interpret_as_same_class

Assumes all pin shapes of a given pin on all cells that with CONN views are of the same *conn_view_eeq_class*, irrespective of the *conn_view_eeq_class* attributes on a pin shape.

-ignore_conn_view_eeq_for_physical_connectivity_tracing

Avoids traversing physical connectivity tracing from one EEQ pin shape of the same *conn_view_eeq_class* to another EEQ pin shape of the same class of the same pin.

-ignore_pins_by_cell_type *cell_type_list*

Specifies the pin instances to be excluded from the physical pin connectivity checking by Milkyway or NDM master cell types. The *cell_type_list* can be a list of Milkyway cell types defined by the *mask_layout_type* attributes, including *antenna*, *chip*, *tap*, *cover*, *macro*, *soft_macro_io_fixed*, *io_pad*, *corner_pad*, *pad_filler*, *flip_chip_pad*, *flip_chip_driver*, *std*, *std_filler*, *double_height_std*, *triple_height_std*, *triple_more_height_std*, and *through_silicon_via*. The cell_type_list also can be a list of NDM cell types defined by the *design_type* attributes, including *analog*, *corner*, *cover*, *diode*, *end_cap*, *fill*, *filler*, *flip_chip_driver*, *flip_chip_pad*, *lib_cell*, *macro*, *module*, *pad*, *pad_spacer*, *physical_only*, *well_tap*.

-ignore_pins_by_cell_type *cell_list*

Specifies the pin instances to be excluded from the physical pin connectivity checking by cell instances. The *cell_list* can be a string list or a cell collection.

-dont_group_contiguous_floating_shapes

Disables to group contiguous floating shapes. By default, the integrity checking engine groups the shapes that are touching, intersecting, or overlapping with each other. The engine considers these contiguous shapes as connected, marks them with the same ID, and reports only one floating error for this group. Using this option to report floating shapes individually.

-missing_via_rule {*layer_name1 route_type1 layer_name2 route_type2*}

Finds possible missing vias with the specified objects. The engine checks only the missing vias between the specified layer names and Milkyway route types. The supported route types include STRAP, RING, MACRO_CONN, STD_CONN, MACRO_PIN, STD_PIN, ALL_WIRE, ALL_PIN, and ALL. Both layer names and route types are case-insensitive. There is no specific order for pairing. For example, {M6 ALl M7 ALL} is equivalent with {M7 ALL M6 ALL}. You can specify multiple layer names and route types at the same time, such as, {{M3 M4 M5} {STD_PIN} {M6 M5 M4} {RING STRAP}}.

-no_stack

Disables stacked via checking. When specified, the engine checks the missing vias between adjacent layers only.

-missing_via_net_shape_direction {*perpendicular (default)* | *parallel* | *both*}

Defines the direction of the net shape objects. The directive states the pair of endpoints are to be checked only if they are either directionally perpendicular, parallel or both.

-merge_same_layer_shapes shape_type_spec_list

Merges all shapes on the same layer that is associated with the shape types defined in the *shape_type_spec_list* when checking for missing vias. The *shape_type_spec_list* is a list of shape types, including *pin_shape*, *net_shap*, *terminal* and *via_metal_enclosure*.

-cut_merged_shape

When the -merge_same_layer_shapes option is specified, the command reports bounding boxes of the merged shapes. However, the merged shapes might result in rectilinear shapes which actually are not rectangular bounding boxes. To retain the original error rectilinear shape, use the -cut_merged_shape option to cut a merged shape into several error rectangles. Each rectangle is large enough to create via on it.

-via_existence_check_partially_enclosed

Allows vias whose cut layers are partially enclosed in the overlap area between the endpoint shapes.

By default, a via's cut layers must be fully enclosed in the overlap area between the endpoint shapes.

This option is valid only when used with the -missing_via_rule option.

-via_existence_check_merge_via_cut

Merges the cut layer shapes of each via array before checking for missing vias and uses the merged virtual via cut layer shape during enclosure checking.

By default, if a single cut of a via array is enclosed within the overlap area, it is considered an existing via.

For information about enclosure checking, see the description of the -via_existence_check_partially_enclosed option.

This option is valid only when used with the -missing_via_rule option.

-ignore_via_enclosure_shape

Removes via enclosure metals from consideration as overlapping endpoints when checking for missing vias.

If you do not merge the enclosure shapes with other layer shapes, use this option to prevent multiple counting of possible missing via errors in the same area.

This option is valid only when used with the -missing_via_rule option.

-existing_stacked_via_check_mode via_cut | via_enclosure

Chooses to detect a stacked via by grouping adjacent layer vias whose via cuts or via metal enclosures overlap with each other. Use the *via_enclosure* argument to detect staircase stacked vias, and the *via_cut* argument for vertically aligned stacked vias.

This option is valid only when used with the -missing_via_rule option.

-allow_stacked_endpoint_vias_outside_of_overlapping_area

Allows the endpoint of a stacked via exceeds the overlapping area during missing via checking. By default, the engine requires the endpoint via cut layers both have a physical connection to the endpoint shapes within the overlap area.

Use the option when checking for staircase stacked vias. This option is valid only when used with the -missing_via_rule option.

-max_via_distance_along_parallel_wires distance

Checks for missing vias only when the distance between separate neighboring vias or stacked vias along the direction of overlapping parallel wires is greater than the given distance value.

This option is valid only when used with the -missing_via_rule option.

-honor_drc_rules

Honors DRC rules to missing via errors. Only limited DRC rules are supported. Missing via errors which violate DRC rules will be filtered out and not reported.

-ignore_minimum_via_criterion

Ignores the criterion that checks if the overlapping area is larger than the via size of the default contact code. When the option is specified, areas that are smaller than the minimum sized vias are not identified as missing vias.

This option is valid only when used with the -missing_via_rule option.

-ignore_routing_blockage_layers *routing_blockage_layer_list*

Ignores all routing blockages associated with the specified blockage layers when checking for missing vias.

By default, the engine honors routing blockages when checking for missing vias. Use this option for what-if analysis. For example, set this option when routing blockages are set for signal routing but they accidentally block supply prerouting vias.

This option is valid only when used the -missing_via_rule option.

-ignore_routing_blockages *blockages*

Ignores all routing blockages associated with the specified blockage objects when checking for missing vias. You can specify the blockages by using a list of blockage object names or a collection returned by the **get_routing_blockages** command.

By default, the engine honors routing blockages. Use this option for what-if analysis.

This option is valid only when used with the -missing_via_rule option.

-ignore_other_pg_net_shapes

Ignores all PG or supply net shapes other than the one specified during analysis when checking for missing vias.

By default, the engine considers PG or supply net shapes are other than the one specified as blockages when checking for missing vias. Use this option for what-if analysis. For example, set this option if the strap location of another supply net is between a supply net's straps and you want to see if more errors arise due to these blocking shapes.

This option is valid only when used with the -missing_via_rule option.

-ignore_small_overlapping_area *area*

Allows the metal overlapping area that is smaller than the given area value. The argument *area* is a floating number in the design unit square.

This option is valid only when used with the -missing_via_rule option.

-filter_layer_net_shapes {{{*layer_name direction*} {*filter_expression*}} ... }

Applies an initial filter for net shapes based on layer name, shape direction, and shape width before checking for missing vias.

layer_name is required and can only be layers specified in the -missing_via_rule option.

direction is optional and can be either < H | V | HV >. The default directionality per layer is HV.

filter_expression is optional and can only include pseudo-attribute *width*, data types integers and floats, and operators: > >= < <= && || ! ().

A net shape is included when the *filter_expression* evaluates to non-zero.

This option does not apply to via enclosure net shapes and CONN view net shapes.

This option has higher precedence than all other -missing_via_rule related options.

Specifically, it is applied before altering geometry through -merge_same_layer_shapes and before applying the filter -missing_via_net_shape_direction.

This option is valid only when used with the -missing_via_rule option.

-report_specific_missing_via_in_stacked_via *true* | *false*

Reports missing via layers in the missing stacked via error. The default value is false.

This option is valid only when used with the -missing_via_rule option.

-enable_discontinuous_connection_check

Detects discontinuous connections for the gaps along the supply or PG routing shapes. If the gap is smaller than the threshold specified with the -discontinuous_connection_threshold option, the engine reports this gap as a discontinuous connection error.

-discontinuous_connection_threshold *metal_layer_gap_distance_pairs*

Specifies the gap distance threshold for each Milkyway routing layer. The *metal_layer_gap_distance_pairs* is a list of *pair_list* elements. Each *pair_list* element consist of two elements; the first is a routing metal layer and the second is a distance value in the library unit. If both explicit layer thresholds and an "ALL" layer threshold is set, the last layer threshold distance value takes precedence over the previous distance settings. The default distance value per layer is the minimum spacing on that layer as specified in the Milkyway technology file.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-ignore_discontinuous_connection_routing_layers *routing_layer_list*

Specifies which routing layers are to be excluded from discontinuous connection checking. By default, the engine checks all Milkyway routing layers for discontinuous connections.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-discontinuous_connection_directionality *metal_layer_directionality_pairs*

Indicates the search direction for discontinuous connection checking. The *metal_layer_directionality_pairs* is a list of *pair_list* elements. Each element consists of two elements; the first is a routing metal layer and the second is a directional setting of either < H | V | HV >. The default directionality per layer is HV.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-ignore_discontinuous_connection_object_types *object_type_spec_list*

Specifies which object types are to be excluded from on discontinuous connection checking. The *object_type_spec_list* is a list of the object *type_spec* items. The valid values are *STRAP*, *RING*, *MACRO_CONN*, fISTD_CONN, *MACRO_PIN*, *STD_PIN*, *IO_PAD_PIN*, *FLIP_CHIP_DRIVER_PIN*, *FLIP_CHIP_PAD_PIN*, *ALL_WIRE*, and *ALL_PIN*.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-ignore_discontinuous_connection_shape_types *shape_type_list*

Specifies which shape types are to be excluded from discontinuous connection checking. The *list_of_shape_types* is a list of Milkyway shape object types. The valid shape types are *pin_shape*, *net_shape*, *via_enclosure* and *terminal*.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-short_circuit_check

Detects short circuit errors between the specified supply nets.

-ignore_other_shape_net_types *shape_net_type_list*

Indicates which shapes are to be excluded from discontinuous connection checking based on the shape's associated net type. The *shape_net_type_list* is a list of Milkyway *net_types*. The valid values are *Signal*, *Power*, and *Ground*. When the option is specified, the engine does not consider the shapes of the specified *net_type* as errors even they blocks the shapes of the same supply net.

This option is valid only when used with the -enable_discontinuous_connection_check option.

-area_edge_condition *touching* | *within* | *intersecting*

Reports errors whose associated area is to be either completely inside of the area with line touching, without line touching or intersecting the boundary of the area.

-included_area_polygon_list {*x1 y1 ... xN yN x1 y1 ...* }

Reports errors whose associated area is inside the specified area.

-excluded_area_polygon_list {*x1 y1 ... xN yN x1 y1 ...* }

Reports errors whose associated area is outside the specified area.

-error_view *prefix_name*

Specifies the name prefix for the Milkyway error cell to be generated. The specified prefix is appended with _<*net_name*>, indicating the supply net for which the error cell applies.

-result *integrity_result_name*

Assigns the name *integrity_result_name* to the integrity result to be generated for saving integrity checking results. When not specified, the command assigns a unique name to the result.

-replace

Overwrites an existing integrity result with the result generated by this command. When used without the -result option, the name assigned by the **current_integrity_result** is used by the new result. When used with the -result option, the specified *rail_result_name* is overwritten with the result generated by this command.

DESCRIPTION

The **check_supply_net_integrity** command analyzes the PG routing associated with a supply net. It checks for potentially missing vias and unconnected geometry segments.

The unconnected geometry check looks for supply net geometries that are electrically floating, that is, not connected to any source of ideal voltage. The **create_taps** command is used to specify ideal voltage sources. The floating geometry report indicates the connected subgroups the floating nets are forming.

The dangling via check looks for dangling vias that are electrically connected but with one open-ended enclosure.

The unconnected pin check looks for PG pins that are electrically floating. When the -pg_pin_floating_connection_check minimal option is specified, the command considers PG pins as not floating if at least one pin shape for each PG or supply pin is physically connected to a supply net that is not floating. When the -pg_pin_floating_connection_check maximal option is specified, the command considers PG pins as not floating if all pin shapes for each PG or supply pin are physically connected to a supply net that is not floating.

The missing via check looks for overlaps of supply net routing on different layers that do not have vias connecting them. If the overlap is large enough to accommodate a minimum-sized via and no via is found in the layer at this location, the command flags it as a missing via.

The area-based filtering options are used to filter out errors outside the specified area. Use the -including_area_polygon_list and -exclused_area_polygon_list options to create a rectilinear area. The command reports errors that are inside the specified area.

If the -error_view option is used to save checking results to an error cell, make sure you have write access to the current Milkyway library using the **set_mw_lib_write_access** command.

EXAMPLES

The following example checks for missing vias, dangling vias, floating shapes, and floating connections:

```
pr_shell> check_supply_net_integrity \e
        -missing_via_rule {ALl ALL ALL ALL} -no_stack \e
        -ideal_sourced_dangling_vias \e
        -non_ideal_sourced_floating_shapes \e
        -pg_pin_floating_connection_check maximal

****************************************
Report : PG geometry integrity report
        -missing_via_rule
        -no_stack
        -ideal_sourced_dangling_vias
        -non_ideal_sourced_floating_shapes
        -pg_pin_floating_connection_check
Design : micro
Version: F-2012.06
Date   : Fri Mar  30 17:20:49 2012
****************************************

Report for supply net: VDD
   Missing vias between layers CP and METAL are 0 total elements
   Missing vias between layers METAL and METAL2 are 2 total elements
   Missing vias between layers METAL2 and METAL3 are 0 total elements
   Missing vias between layers METAL3 and METAL4 are 2 total elements
   Missing vias between layers METAL4 and METAL5 are 52 total elements
   Connectivity check result:
      There are 3 connected net groups.
      There are 5 unconnected net groups.
      There are 4 dangling via(s).
      There are 1 unconnected pins.
```

The following example checks for missing vias between M1 and M3, M2 and M3, after first filtering horizontal net shapes on layer M1 of width > 0.1 units and vertical net shapes on layer M2 of width < 0.1 units.

```
pr_shell> check_supply_net_integrity \e
        -missing_via_rule {{M1 M2} ALL {M3} ALL} \e
        -filter_layer_net_shapes {{{M1 H} {width>0.1}} \e
                                  {{M2 V} {width<0.1}}}
```

SEE ALSO

create_taps(2)
current_integrity_result(2)
remove_integrity_result(2)
rename_integrity_result(2)
report_integrity_results(2)
report_taps(2)
set_mw_lib_write_access(2)

check_taps

Checks for valid locations of tap points that are to be used in subsequent rail analysis. In NDM flow, the validation of tap locations is embedded inside the **create_taps** command and hence **check_taps** is disabled.

SYNTAX

```
status check_taps
   [supply_nets]
```

Data Types

```
list supply_nets
```

ARGUMENTS

supply_nets

Lists of supply net specification elements for which tap location validity is to be checked. Each element in the supply_nets list can be a supply net collection or name pattern. The list of supply nets is automatically expanded to include the supply nets that are connected to those listed through MTCMOS switch cells.

DESCRIPTION

The **create_taps** command checks validity for tap point's location created in the current session that are associated with the supply_nets argument and the is_valid attribute setting is unknown. Also, by default, **create_taps** checks the validity of each tap point's location during each invocation. If there are tap points created using a previous command *create_taps* with the option *-nocheck*, those taps will not be checked upon creation and the is_valid attribute will be set to unknown. In this situation use the **check_taps** command. If all tap points in the current session are already checked using the command **create_taps**, running the **check_taps** command only checks if there is at least one valid tap point for each non switched supply net in the list. If there are no valid tap points found for a non-switched supply net in the list, the command will return a failure condition. This return information is useful to determine if a non-switched supply net has the minimal sufficient tap setting information for the subsequent rail or integrity analysis.

EXAMPLES

This example creates a tap point using the **create_taps** command with the -nocheck option, so its location's validity is not checked.. The subsequent **check_taps** command then checks its location's validity for the tap point

and the supply net. In addition, a summary of the count of all tap's location validity for each supply net is reported. This summary shows the taps location was determined to be invalid.

```
pr_shell> create_taps -layer 3 -point {100.0 200.0} -supply_nets VDD -nocheck
1
pr_shell> check_taps VDD
Net            Valid           Invalid
--------------------------------------------------------------------------------
VDD            0               1
--------------------------------------------------------------------------------
0
```

The following example creates a tap point using the **create_taps** command with its built-in validity check since the -nocheck option is not specified. The subsequent **check_taps** command only checks if there is at least one valid tap point for the supply net (which there is in this case) and returns a success code of 1.

```
pr_shell> create_taps -layer 5 -point {200 300} -supply_nets VDD
1
pr_shell> check_taps VDD
Net            Valid           Invalid
--------------------------------------------------------------------------------
VDD            1               0
--------------------------------------------------------------------------------
1
```

SEE ALSO

create_taps(2)

check_timing

Reports possible timing problems in the design.

SYNTAX

```
string check_timing
   [-verbose]
   [-significant_digits digits]
   [-ms_min_separation delta]
   [-override_defaults check_list]
   [-include check_list]
   [-exclude check_list]
```

Data Types

```
digits          int
delta           float
check_list      list
```

ARGUMENTS

-verbose

Shows detailed information about potential problems.

-significant_digits *digits*

Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13. The default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option if you want to override the default.

-ms_min_separation *delta*

Minimum separation value between master and slave clocks. The default minimum separation is 0.0.

-override_defaults *check_list*

Overrides the checks in the **timing_check_defaults** variable using the *check_list* option. For default information, see the **timing_check_defaults** man page.

-include *check_list*

Adds the checks specified in the *check_list* to the checks in **timing_check_defaults** variable.

-exclude *check_list*

Subtracts the checks specified in the *check_list* from the checks in **timing_check_defaults** variable.

DESCRIPTION

Checks the assertions and structure of the design for potential timing violations. This command is used to identify possible problems before generating timing or constraint reports.

This command also prints which checks it performs. If a check reveals a violation, the command also prints a message about the violation. By default, the message contains a summary of the violation. To get more information about violations, use the **-verbose** option.

This command without any options performs the checks defined by the **timing_check_defaults** variable. To change the value of this variable, either redefine it or use the **-override_defaults** option. If the **-override_defaults** option is not used, the final list of checks to be performed is the checks in the **timing_check_defaults** variable plus the list of checks given by the **-include** option minus the list of checks given by the **-exclude** option. If the **-override_defaults** option is used with a *check_list*, the final list of checks to be performed is the checks in the *check_list* given by the option plus the list of checks given by the **-include** argument minus the list of checks given by the **-exclude** argument.

You can specify the following checks in the *check_list* argument:

- **clock_crossing** - Checks clock interactions when there are multiple clock domains. If a clock launches one or more paths, which are captured by other clocks, it is listed in clock crossing report. If all paths between two clocks are false paths or they are exclusive/asynchronous clocks, the path is marked by *. If only part of paths are set as false paths or exclusive/asynchronous clocks, the path is marked by #.
- **data_check_multiple_clock** - Warns if multiple clocked signals reach a data check register reference pin. The analysis is performed for each of the clocked domain separately.
- **data_check_no_clock** - Warns if no clocked signal reaches a data check register reference pin. In this case, no setup or hold checks are performed on the constrained pin.
- **generated_clocks** - Checks generated clock network. The master must be driven by a clock source. There cannot be loops of generated clocks. For example, the source of generated clock CLK1 cannot be used to generate clock CLK2 if CLK2 also is used to generate CLK1.
- **generic** - Warns about generic (unmapped) cells in the design. The timing of paths through generic cells is inaccurate because generic cells have zero delay.
- **ideal_clocks** - Shows the clocks that are not defined as propagated using the **set_propagated_clock** command. Generally, all clocks should be propagated so that the clock network timing is accurately calculated. Especially, in presence of crosstalk, the delay changes induced by other nets on the clock network are not reflected in the calculated slacks in the design.
- **latch_fanout** - Checks fanout of level-sensitive latches. A warning is issued if a level-sensitive latch fans out to itself. An information message also appears for a latch that fans out to a latch of the same clock (PHI1 to PHI1 path).
- **latency_override** - Warns of clock latency specification conflicts. If clock source latency is defined for both a clock and its port (source pin), the source latency for clock object is ignored. If input_delay is set on clock port, which also has source latency specified, the input_delay is ignored as a source latency. Also warns if more than one clock latency fan out to any latch clock pin.
- **loops** - Warns of combinational feedback loops. Combinational feedback loops are not analyzed. If the feedback loop is not broken by **set_disable_timing**, it is automatically broken by disabling one or more timing arcs.
- **ms_separation** - Checks minimum separation of master and slave clock pulses on master/slave latches.

- **no_clock** - Warns if no clock reaches a register clock pin. In this case, no setup or hold checks are performed on data pins related to that clock pin, and the path starting at the clock pin is not relative to a clock.
- **no_driving_cell** - Warns if a port does not have any driving cell. This warning is issued only when the net connected to the port has parasitics. In such case, the accuracy of delay calculation could be affected, as a default strong driver is assumed in absence of driving cell definition. Especially, in presence of crosstalk, a port with no driving cell could act as a strong aggressor which could lead to significant amount of pessimism in the analysis. Also, a port with no driving cell could act as a string victim, which could underestimate the crosstalk effect.
- **no_input_delay** - Warns if no clock related delay specified on an input port, where it propagates to a clocked latch or output port. With **-verbose**, the port name is listed. Note that with **timing_input_port_default_clock** set to true, a default clock is assumed for the input port. Otherwise it is not be clocked, and the paths are unconstrained. In this case, if there is no input delay specified, check_timing does not generate warnings.
- **operating_conditions** - When the design has scaling groups defined by the **define_scaling_lib_group** command, this operating condition check detects improperly set process, voltage, temperature, rail, and operating conditions. Warning are issued for
  - Extrapolation outside the PVT rail operating condition range covered by the scaling group.
  - If **-exact_match_only** is specified in the **define_scaling_lib_group** command, it warns on exact-match failures when the design cell instance operating condition does not match with any of the corner library operating condition.
  - When a design cell instance does not have an associated scaling group, it warns when the operating condition does not match with that of the linked library operating condition.
- **partial_input_delay** - Warns if any port has partially defined input delay. This happens when **set_input_delay -min** is applied on a port to set the min input delay with respect to a clock, however no **set_input_delay -max** is applied to that port to specify the max delay, or vice versa. As a result, some paths starting from the port with partially defined input delay might become unconstrained and some potential violations could be missed.
- **pll_configuration** - Warns if a problem is detected in the configuration of any phase locked loop (PLL) cells. For a PLL to be correctly configured, the PLL output clock should reach the PLL feedback pin. If a PLL is not correctly configured, it does not show any phase adjustment on the PLL output clock.
- **pulse_clock_no_pulse_generator** - Warns if the pulse clock constraints cannot be honored. This could be because pulse clock constraints have been set on clocks that do not drive pulse generators, or the design might not have any pulse generators.
- **pulse_clock_non_pulse_clock_merge** - Warns if pulse clock and normal clock propagate to the same network.
- **retain** - Warns if any of the RETAIN values is larger than its corresponding delay value. The RETAIN values should be less than their corresponding delay values so that they be considered for hold violations. Otherwise, they might be considered for setup violations.
- **signal_level** - Checks that the driver signal level matches the load signal level. The signal levels are determined from the cell specific operating conditions and rail voltages (or UPF), and from the following library attributes: input_signal_level, output_signal_level, input_voltage, output_voltage. The check is performed on all input pins that have input_voltage defined in the timing library. If the driver pin does not have output_voltage defined in the library then the voltage of the rail powering the driver pin is used as the output signal level. With PG libraries the signal level is determined from input/output_signal_level_high (exact voltage in .lib) first, then input/output_signal_level (reference to voltage_map in .lib) if the related_power_pin uses a different_voltage map, and finally from the instance specific voltage of the related power PG pin. If the

load pin does not have input_voltage then matching is done based on driver and load rail voltages. Such matching (without library attributes input/output_voltage) is subject to tolerances set by **set_level_shifter_threshold** and **set_level_shifter_strategy** commands. By default the number of mismatches is reported. With **-verbose** the mismatching driver, load and voltage difference are reported.

- **supply_net_voltage** - Checks that each segment of UPF supply nets has voltage assigned to it by **set_voltage** command. It warns about mismatching voltage between port supply voltage against port driving cell voltage. It also warns about mismatching port supply against visible logic. The port supply refers to port supply attributes specified on port objects through the **set_port_attributes** command.
- **unconnected_pg_pins** - Checks that each power and ground pin is connected to a UPF supply net. The connection can be implicit (for example, power domain) or explicit (for example, **connect_supply_net**).
- **unconstrained_endpoints** - Warns about unconstrained timing endpoints. This warning identifies timing endpoints (output ports and register data pins) that are not constrained for maximum delay (setup) checks. If the endpoint is a register data pin, it can be constrained by using **create_clock** for the appropriate clock source. You can constrain output ports using the **set_output_delay** or **set_max_delay** commands.
- **unexpandable_clocks** - Warns if there are sets of clocks for which periods are not expandable with respect to each other. The checking is only done for the related clock domains, such as ones where there is at least one path from one clock domain to the other. This could be because of an incorrectly defined clock period for one or more of the clocks. Another possibility is when asynchronous clocks with unexpandable periods are interacting where they should have been defined in different clock domains.

EXAMPLES

The following example checks timing of the current design and shows summary information of potential problems. The checks performed are those defined by **timing_check_defaults**.

```
pt_shell> check_timing

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.
```

The following example adds the **clock_crossing** check to the list of checks in **timing_check_defaults**.

```
pt_shell> check_timing -include {clock_crossing}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.
```

```
Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.

Information: Checking 'clock_crossing'.

Information: Checking 'pll_configuration'.
```

The following example adds the **clock_crossing** check to the list of checks in **timing_check_defaults** and removes the **loops** check from the same list.

```
pt_shell> check_timing -include {clock_crossing} -exclude {loops}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'generated_clocks'.

Information: Checking 'clock_crossing'.
```

The following example removes the **retain** option from the list of checks in **timing_check_defaults**. Assuming that this check is not included in **timing_check_defaults**, there is nothing to remove. The output in this example is the same as running the command without the **-exclude** option.

```
pt_shell> check_timing -exclude {retain}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum delay.

Information: Checking 'generic'.
```

```
Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.
```

The following example checks only latch fanout and shows detailed information.

```
pt_shell> check_timing -verbose -override_defaults {latch_fanout}
Warning: There are 2 level-sensitive latches which fanout to themselves.

Latch data pin
------------------------------------------------------------
l3/D
l4/D

Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

From Pin            To Pin              Clock         Level
--------------------------------------------------------------------------------
l1/G                l2/D                C1            positive
l1/G                l3/D                C1            positive
```

SEE ALSO

create_clock(2)
report_constraint(2)
report_timing(2)
set_clock_groups(2)
set_case_analysis(2)
set_disable_timing(2)
set_false_path(2)
set_max_delay(2)
set_output_delay(2)
create_power_domain(2)
create_supply_net(2)
timing_check_defaults(3)
timing_input_port_default_clock(3)
report_default_significant_digits(3)

close_mw_cel

Closes the currently open Milkyway design.

SYNTAX

```
status close_mw_cel
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

Closes the open Milkyway design that has been read in by the **open_mw_cel** command. This command removes the current design and all generated data from the session memory. The generated rail results are no longer be available after the **close_mw_cel** command is executed.

EXAMPLES

The following example saves a session and then removes the most recently loaded Milkyway design and its associated rail results from the current session:

```
pr_shell> current_rail_result
vdrop_rail_result


pr_shell> close_mw_cel
1

pr_shell> current_rail_result
none
```

SEE ALSO

open_mw_cel(2)

```
current_rail_result(2)
```

close_mw_lib

Closes the current Milkyway library.

SYNTAX

```
status close_mw_lib
   [-save]
```

ARGUMENTS

-save

Saves all Milkyway CEL views opened in this library. By default, the command discards changes of Milkyway CEL views in this library.

DESCRIPTION

This command closes the current Milkyway library. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example closes the current Milkyway library:

```
prompt> close_mw_lib
1
```

SEE ALSO

open_mw_lib(2)

collections

Describes the methodology for creating collections of objects and querying objects in the database.

DESCRIPTION

Synopsys applications build an internal database of objects and attributes applied to them. These databases consist of several classes of objects, including designs, libraries, ports, cells, nets, pins, clocks, and so on. Most commands operate on these objects.

By definition:

A collection is a group of objects exported to the Tcl user interface.

Collections have an internal representation (the objects) and, sometimes, a string representation. The string representation is generally used only for error messages.

A set of commands to create and manipulate collections is provided as an integral part of the user interface. The collection commands encompass two categories: those that create collections of objects for use by another command, and one that queries objects for viewing. The result of a command that creates a collection is a Tcl object that can be passed along to another command. For a query command, although the visible output looks like a list of objects (a list of object names is displayed), the result is an empty string.

An empty string "" is equivalent to the empty collection, that is, a collection with zero elements.

To illustrate the usage of the common collection commands, the man pages have examples. Most of the examples use PrimeTime as the application. In all cases, the application from which the example is derived is indicated.

Homogeneous and Heterogeneous Collections

A homogeneous collection contains only one type of object. A heterogeneous collection can contain more than one type of object. Commands that accept collections as arguments can accept either type of collection.

Lifetime of a Collection

Collections are active only as long as they are referenced. Typically, a collection is referenced when a variable is set to the result of a command that creates it or when it is passed as an argument to a command or a procedure. For example, in PrimeTime, you can save a collection of design ports by setting a variable to the result of the **get_ports** command:

```
pt_shell> set ports [get_ports *]
```

Next, either of the following two commands deletes the collection referenced by the *ports* variable:

```
pt_shell> unset ports
pt_shell> set ports "value"
```

Collections can be implicitly deleted when they go out of scope. Collections go out of scope for various reasons. An example would be when the parent (or other antecedent) of the objects within the collection is deleted. For example, if our collection of ports is owned by a design, it is implicitly deleted when the design that owns the ports is deleted. When a collection is implicitly deleted, the variable that referenced the collection still holds a

string representation of the collection. However, this value is useless because the collection is gone, as illustrated in the following PrimeTime example:

```
pt_shell> current_design
{"TOP"}

pt_shell> set ports [get_ports in*]
{"in0", "in1"}

pt_shell> remove_design TOP
Removing design 'TOP'...

pt_shell> query_objects $ports
Error: No such collection '_sel26' (SEL-001)
```

Iteration

To iterate over the objects in a collection, use the **foreach_in_collection** command. You cannot use the Tcl-supplied **foreach** iterator to iterate over the objects in a collection, because the **foreach** command requires a list, and a collection is not a list. In fact, if you use the **foreach** command on a collection, it destroys the collection.

The arguments of the **foreach_in_collection** command are similar to those of foreach: an iterator variable, the collection over which to iterate, and the script to apply at each iteration. Note that unlike the **foreach** command, the **foreach_in_collection** command does not accept a list of iterator variables.

The following example is an iterative way to perform a query in PrimeTime. For more information, see the **foreach_in_collection** man page.

```
pt_shell> \
foreach_in_collection s1 $collection {
  echo [get_object_name $s1]
  }
```

Manipulating Collections

A variety of commands are provided to manipulate collections. In some cases, a particular command might not operate on a collection of a specific type. This is application-specific. Consult the man pages from your application.

- **add_to_collection** - This command creates a new collection by adding a list of element names or collections to a base collection. The base collection can be the empty collection. The result is a new collection. In addition, the **add_to_collection** command allows you to remove duplicate objects from the collection by using the *-unique* option.
- **append_to_collection** - This command appends a set of objects (specified by name or collection) to an existing collection. The base collection is passed in through a variable name, and the base collection is modified directly. It is similar in function to the **add_to_collection** command, except that it modifies the collection in place; therefore, it is much faster than the **add_to_collection** command when appending.
- **remove_from_collection** - This command removes a list of element names or collections from an existing collection. The second argument is the specification of the objects to remove and the first argument is the collection to have them removed from. The result of the command is a new collection. For example, in PrimeTime:

  ```
  pt_shell> set dports \
  [remove_from_collection [all_inputs] CLK]
  {"in1", "in2", "in3"}
  ```

- **compare_collections** - This command verifies that two collections contain the same objects (optionally, in the same order). The result is "0" on success.

- **copy_collection** - This command creates a new collection containing the same objects in the same order as a given collection. Not all collections can be copied.
- **index_collection** - This command extracts a single object from a collection and creates a new collection containing that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index. Not all collections can be indexed.
- **sizeof_collection** - This command returns the number of objects in a collection.

Filtering

In some applications, you can filter any collection by using the **filter_collection** command. This command takes a base collection and creates a new collection that includes only those objects that match an expression.

Some applications provide a *-filter* option for their commands that create collections. This allows objects to be filtered out before they are ever included in the collection. Frequently this is more efficient than filtering after the they are included in the collection. The following examples from PrimeTime filters out all leaf cells:

```
pt_shell> filter_collection \
[get_cells *] "is_hierarchical == true"]
{"i1", "i2"}
pt_shell> get_cells * -filter "is_hierarchical == true"
{"i1", "i2"}
```

The basic form of a filter expression is a series of relations joined together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. In the previous example, *is_hierarchical* is the attribute, == is the relational operator, and *true* is the value.

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only true or false.

Additionally, existence relations determine if an attribute is defined or not defined, for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class. See the appropriate man pages for complete details.

Sorting Collections

In some applications, you can sort a collection by using the **sort_collection** command. It takes a base collection and a list of attributes as sort keys. The result is a copy of the base collection sorted by the given keys. Sorting is ascending, by default, or descending when you specify the *-descending* option. In the following example from PrimeTime, the command sorts the ports by direction and then by full name.

```
pt_shell> sort_collection [get_ports *] \
{direction full_name}
{"in1", "in2", "out1", "out2"}
```

Implicit Query of Collections

In many applications, commands that create collections implicitly query the collection when the command is used at the command line. Consider the following examples from PrimeTime:

```
pt_shell> set_input_delay 3.0 [get_ports in*]
1
pt_shell> get_ports in*
{"in0", "in1", "in2"}
pt_shell> query_objects -verbose [get_ports in*]
{"port:in0", "port:in1", "port:in2"}
pt_shell> set iports [get_ports in*]
{"in0", "in1", "in2"}
```

In the first example, the **get_ports** command creates a collection of ports that is passed to the **set_input_delay** command. This collection is not the result of the primary command (**set_input_delay**), and as soon as the primary command completes, the collection is destroyed. The second example shows how a command that creates a collection automatically queries the collection when that command is used as a primary command. The third example shows the verbose feature of the **query_objects** command, which is not available with an implicit query. Finally, the fourth example sets the variable iports to the result of the **get_ports** command. Only in the final example does the collection persist to future commands until iports is overwritten, unset, or goes out of scope.

SEE ALSO

add_to_collection(2)
append_to_collection(2)
compare_collections(2)
copy_collection(2)
filter_collection(2)
foreach_in_collection(2)
index_collection(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)
sort_collection(2)

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.

SYNTAX

```
int compare_collections
   [-order_dependent]
   collection1
   collection2
```

Data Types

collection1	collection
collection2	collection

ARGUMENTS

-order_dependent

Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.

collection1

Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2

Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

DESCRIPTION

The **compare_collections** command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of cells u1 and u2 is the same as a collection of the cells u2 and u1. By using the **-order_dependent** option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (that is, **compare_collections** considers them identical), and the result is "0".

EXAMPLES

The following example from PrimeTime shows a variety of comparisons. Note that a result of "0" from **compare_collections** indicates success. Any other result indicates failure.

```
pt_shell> compare_collections [get_cells *] [get_cells *]
0
pt_shell> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
pt_shell> set c2 [get_cells {u2 u1}]
{"u2", "u1"}
pt_shell> set c3 [get_cells {u2 u4 u6}]
{"u2", "u4", "u6"}
pt_shell> compare_collections $c1 $c2
0
pt_shell> compare_collections $c1 $c2 -order_dependent
-1
pt_shell> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
pt_shell> set c4 ""
pt_shell> compare_collections $c1 $c4
-1
pt_shell> compare_collections $c4 $c4
0
```

SEE ALSO

collections(2)

compare_interface_timing

Compares two reports generated by the **write_interface_timing** command.

SYNTAX

```
int compare_interface_timing
   ref_timing_file
   cmp_timing_file
   [-output file_name]
   [-absolute_tolerance atol_list]
   [-percent_tolerance ptol_list]
   [-capacitance_tolerance ctol_list]
   [-noise_tolerance ntol_list]
   [-ignore ign_list]
   [-include cmp_list]
   [-quiet]
   [-nosplit]
   [-sort_by_worst]
   [-session session_name]
   [-significant_digits digits]
```

Data Types

ref_timing_file	string
cmp_timing_file	string
file_name	string
atol_list	list
ptol_list	list
ctol_list	list
ntol_list	list
ign_list	list
cmp_list	list
session_name	string
digits	int

ARGUMENTS

ref_timing_file

Specifies the name of the timing file to be used as the reference in the comparison.

cmp_timing_file

Specifies the name of the timing file to be compared to the reference timing file.

-output *file_name*

Specifies the name of the output file where the results of the comparison information is written. The information

written to *file_name* specifies PASS/FAIL status for every parameter compared by the command.

By default, these results are written to the session transcript. All informational, warning, or error messages are still directed to the session transcript, even if this option is specified.

-absolute_tolerance *atol_list*

Specifies the absolute error tolerance for time data. The default is zero. See below for a description of the tolerance format. If you specify this option together with the **-percent_tolerance** option, both tolerances must be exceeded to cause a FAIL. See the **-capacitance_tolerance** and **-noise_tolerance** options for other absolute tolerances.

-percent_tolerance *ptol_list*

This option is similar to the **-absolute_tolerance** option except that it uses the percent relative error instead of an absolute error. The exception is that slack data ignores this option and uses only the absolute tolerance. The default is zero.

See below for a description of the tolerance format. If you specify this option together with either the **-absolute_tolerance**, **-capacitance_tolerance** or **-noise_tolerance** option, both tolerances must be exceeded to cause a FAIL.

-capacitance_tolerance *ctol_list*

Specifies the absolute error tolerance for capacitance data. The default is zero.

See below for a description of the tolerance format. If you specify this option together with the **-percent_tolerance** option, both tolerances must be exceeded to cause a FAIL. For other absolute tolerances, see the **-absolute_tolerance** and **-noise_tolerance** options.

-noise_tolerance *ntol_list*

Specifies the absolute error tolerance for noise slack. The default is zero.

See below for a description of the tolerance format. If you specify this option together with the **-percent_tolerance** option, both tolerances must be exceeded to cause a FAIL. For other absolute tolerances, see the **-absolute_tolerance** and **-capacitance_tolerance** options.

-ignore *ign_list*

Specifies a list of categories of slack or arc data to be excluded from the comparison. The following list describes the valid values:

max - Excludes all max_rise and max_fall delay type arcs from the slack and transition_time sections. The capacitance section is not examined because it contains only maximum information.

min - Excludes all min_rise and min_fall delay type arcs from the slack and transition_time sections.

unconstrained - Excludes all paths that are unconstrained for both reference and compare data files.

pass - Excludes all passing comparisons, so only fails are displayed.

async_default - Excludes the comparison of paths in the async_default path group. This option is obsolete and is replaced by specifying the **recover** and **removal** values of the **-include** option.

clock_gating - Excludes comparison of paths with the clock_gating_default path group. This option is obsolete and is replaced by specifying the **clock_gating_setup** and **clock_gating_hold** of the **-include** option.

input_to_register - Excludes comparison of input-to-register paths. This option is obsolete and is replaced by specifying the **setup** and **hold** values of the **-include** option.

register_to_output - Excludes comparison of register-to-output path. This option is obsolete and is replaced by specifying the **max_seq_delay** and **min_seq_delay** of the **-include** option.

input_to_output - Excludes comparison of combinational paths. This option is obsolete and is replaced by specifying the **max_combo_delay** and **min_combo_delay** of the **-include** option.

-include *cmp_list*

Specifies a list of data sections from the **write_interface_timing** command output to be included in the comparison; only those sections are compared. By default, all sections are compared. The following list describes the keywords that are accepted in the *cmp_list* as valid:

slack - Specifies that only worst-case slacks are to be compared.

arc - Specifies that only arc values are to be compared.

transition_time - Specifies that only actual transition times on ports are to be compared.

capacitance - Specifies that only actual capacitances on ports are to be compared.

design_rules - Specifies that only design rules on ports are to be compared.

missing_arcs - Specifies that only arcs in the reference timing file that are missing from the compare timing file are to be included in the report. If it is used together with other sections, the report includes the intersection of **missing_arcs** and the other specified sections, not the union of them.

max_combo_delay - Specifies that only maximum combination delay arc type paths are to be reported.

min_combo_delay - Specifies that only minimum combination delay arc type paths are to be reported.

max_seq_delay - Specifies that only maximum sequential delay arc type paths are to be reported.

min_seq_delay - Specifies that only minimum sequential delay arc type paths are to be reported.

setup - Specifies that only setup arc type paths are to be reported.

hold - Specifies that only hold arc type paths are to be reported.

recovery - Specifies that only recovery arc type paths are to be reported.

removal - Specifies that only removal arc type paths are to be reported.

clock_gating_setup - Specifies that only clock gating setup arc type paths are to be reported.

clock_gating_hold - Specifies that only clock gating hold arc type paths are to be reported.

noise - Specifies that the noise section should be included.

-quiet

Specifies an obsolete option. This option is now obsolete and has been replaced by the **-session quiet** option.

-nosplit

Prevents line-splitting. This is most useful for performing difference comparisons on previous scripts or for post processing the script.

-sort_by_worst

Specifies that the output is to be sorted from negative to positive, with the worst failure first. After the FAIL section, all PASS lines are sorted alphabetically within each path attribute. By default, the output follows the order of the reference write_interface_timing data, which is sorted alphabetically within each path attribute (c paths, i paths including a and g, then o paths).

-session ***session_name***

Controls the information that is printed to the session transcript. Note that this option affects only the session transcript, not the information written to the output file. The following list describes the keywords that are accepted for *session_name*:

quiet - Prevents information, warning, and error messages from appearing in the session transcript. Use this option when you want to see only the success/failure return code without any MV messages. The **-session** option used with this keyword replaces the now-obsolete **-quiet** option.

summary - Prints only the summary pass/fail count to the session transcript.

full - Prints the detailed report either to the session transcript or to a file if you specify the **-output** option. This is the default behavior.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the **report_timing** command. Allowed values are 0-13. The default is 2. Use this option if you want to override the default. See below for a more detailed description of significant digits.

DESCRIPTION

This command compares two interface timing files (called the "reference" and "comparison" files) previously generated by the **write_interface timing** command. The result is PASS if the timing parameter values in the two files are the same or within a specified tolerance. The result is FAIL if both columns have data and the tolerance is exceeded. If all comparisons in the report are PASS, the return code for the **compare_interface_timing** command is 0. If one or more comparisons result in FAIL, the return code is 1. Any other return code (or no return code) indicates a command error. If either file format does not conform to the **write_interface_timing** standard, the command issues a warning message.

The **compare_interface_timing** command lets you specify the input and output files for the comparison, the types of paths and timing parameters to compare or not compare, and the allowed tolerance levels that trigger comparison failures.

If the reference and comparison files were generated using different contexts then the two files might contain different timing arcs along with different slacks. If an arc exists in the comparison file but not in the reference file, the **compare_interface_timing** command ignores the extra arcs. If an arc exists in the comparison file but not in the reference file, the **compare_interface_timing** command reports the slack value for the reference file, but either "N.C." or "----" for the comparison file slack. The slack difference is marked as "----", and the status is FAIL. The comparison slack is marked as "----" if the arc is not clock gating or recovery or removal.

The slack for a missing arc is mark as "N.C.", meaning not critical, if it exists in the reference but not in the comparison file, and the arc type is clock gating or recovery or removal. These arcs fall into the **clock_gating_default** or **asynchronous_default** clock groups, respectively. Each of these groups can have several clocks in them, and under certain circumstances the **write_interface_timing** command sees the path to one clock as critical for the reference view and report that arc and slack, but another clock as critical in the comparison view and report that arc and slack.

The **compare_interface_timing** command deals with significant digits in two distinct ways. First, the internal computations are limited to the minimum precision of the two input files. This means that if the *ref_timing_file* was generated with three digits and the *cmp_timing_file* with four, the **compare_interface_timing** command uses three digits for all internal computations.

It is important to make the tolerances larger than a unit of the computational significant digits, or the

compare_interface_timing command could generate false FAIL and PASS messages. This can happen because of rounding. For example, if the internal significant digits is two, and the absolute tolerance is set to 0.01, a difference of 0.014 is rounded to 0.01 and reported as a PASS even though it is outside the tolerance. Similarly, if the absolute tolerance is set to 0.009, a difference of 0.009 is rounded to 0.01 and reported as FAIL even though it is within tolerance. You should ensure that the number of digits of accuracy of the input files is greater than the accuracy of the tolerances.

A second use of significant digits is in report generation, which is limited to the minimum of the input precision and the significant digits that you specified. You specify the significant digits for the report through the **-significant_digits** option. For example, if the value for the **-significant_digits** option is five but the *ref_timing_file* was generated with four digits, the report is limited to four digits. Note that this use of significant digits does not affect the PASS or FAIL status, but a reported difference could look like it should have a different status if the difference is close to a tolerances and the difference rounded when writing the report.

Each tolerance is a list of either one or two floating point numbers. All tolerances are inclusive, and the sign of the values has no effect. If there is only one number, it serves to specify both the plus and minus tolerances. The plus tolerance is the absolute value of the number, and the minus tolerance is the inverse of the plus. For example, *atol_list* equal to **0.1** indicates that time differences of less than 0.1 and greater than -0.1 are considered to be passing.

If there are two numbers, the first specifies the minus bound and the second the plus. The minus tolerance is the inverse of the absolute value of the first number, and the plus tolerance is the absolute value of the second number. Having a pair of numbers allows different tolerances to be applied for negative and positive errors, which can be interpreted as optimism and pessimism depending on the arc type. For example, an *atol_list* equal to **{0.2 -0.3}** indicates that data differences of less than 0.3 and greater than -0.2 are considered passing.

EXAMPLES

The following example shows a variety of comparisons. Each item in parentheses (for example, L1, L2) represents the latch level associated with the slack number directly to the left. These values are included in the report wherever there is an **L#** attribute in the **write_interface_timing** output. At the end of the report is a summary of the total comparisons passed and failed broken down by data section.

```
pt_shell> compare_interface_timing demo_net.rpt demo_etm.rpt \
            -include arc -percent_tolerance {10 5}

****************************************
Command: compare_interface_timing
         demo_net.rpt demo_etm.rpt
         -include arc
         -percent_tolerance 10.0 5.0
Design : top
 ...
****************************************

                Arc                          Arc Value
From    To      Type                     Ref             Cmp    %Error Status
-------------------------------------------------------------------------------
in(r)   out(r)  max_combo_delay          2.63            2.21    15.97   FAIL
in(f)   out(r)  max_combo_delay          2.28            2.32    -1.75   PASS
in(r)   out(f)  min_combo_delay          0.17            0.30   -76.47   FAIL
in(f)   out(f)  min_combo_delay          0.51            0.51     0.00   PASS
in(r)   clk(r)  setup                    2.63    (L1)    2.63     0.00   PASS
in(f)   clk(r)  setup                    2.29    (L2)    2.29     0.00   PASS
in(r)   clk(f)  hold                     0.17            0.17     0.00   PASS
clk(r)  out(r)  max_seq_delay           17.79           17.79     0.00   PASS
clk(r)  out(f)  max_seq_delay           18.20           18.20     0.00   PASS
```

```
clk(f)  out(r)  min_seq_delay          2.21            2.21     0.00   PASS
clk(f)  out(f)  min_seq_delay          1.80            1.80     0.00   PASS


        Totals  Arc Value Transition Time   Capacitance  Rules
-----------------------------------------------------------------
Passed  9       9         0                 0            -
Failed  2       2         0                 0            0
Total   12      12        0                 0            0
```

SEE ALSO

write_interface_timing(2)

complete_net_parasitics

Completes partial parasitics annotated on all nets of the current design.

SYNTAX

```
string complete_net_parasitics
   [-complete_with completion_type]
```

Data Types

completion_type	string

ARGUMENTS

-complete_with *completion_type*

Indicates that a net with partially annotated parasitics and missing segments is to be completed by inserting capacitances and resistances according to the *completion_type* value. Allowed values are *zero* (the default), which completes the net by inserting zero capacitances and resistances, and *wlm*, which completes the net by inserting capacitances and resistances derived from wire load models.

This option is equivalent to the **read_parasitics -complete_with** command.

DESCRIPTION

This command completes all nets in the design that have incomplete RC networks annotated from SPEF or SPF files.

If lumped parasitics (set using the **set_load** or **set_resistance** command) already exist on any of the nets, they are not overwritten; however, a warning message is issued.

Note: The **complete_net_parasitics** and **read_parasitics -complete_with** commands complete a net only if all missing segments are between two pins, and only if the nets are partially annotated. Nets are not affected if they are fully annotated or have no annotation at all. Also, the net must be hierarchical, so that if the parasitics for the block-level parts of a net are missing, those parasitics could exist in the top-level net. If any of these conditions are not met, you must manually correct the SPEF or DSPF file.

Use the **report_annotated_parasitics** command to view how the parasitics are completed.

After the completion of the nets, there is no direct way to remove only the completed segments. You can remove all parasitics read and annotated with the **read_parasitics** and **complete_net_parasitics** commands using the **remove_annotated_parasitics** command. You then reread the previously annotated parasitics using the

read_parasitics command. The **reset_design** command removes all attributes from the current design, including annotated parasitics.

EXAMPLES

The following example completes the partially annotated parasitics from the file speffile.spef, by inserting zero capacitances and resistances.

```
pt_shell> read_parasitics speffile.spef
pt_shell> complete_net_parasitics -complete_with zero
```

SEE ALSO

```
read_parasitics(2)
remove_annotated_parasitics(2)
report_annotated_parasitics(2)
reset_design(2)
```

connect_pg_net

Connects the specified pins of the cell instances to supply nets.

SYNTAX

```
connect_pg_net
   -cells cells
   -power_pin pin_name
   -power_net supply_net_name
   -ground_pin pin_name
   -ground_net supply_net_name
   [-remove]
   [-verbose]
```

Data Types

```
collection cells
string pin_name
string supply_net_name
```

ARGUMENTS

cells

Lists the cell instances to be connected. Each element of the list can be a cell instance collection or a name pattern.

-power_pin *pin_name*

Specifies the name of the power pin to connect to the supply net.

-power_net *supply_net_name*

Specifies the name of the power supply net to be connected.

-ground_pin *pin_name*

Specifies the name of the ground pin to connect to the supply net.

-ground_net *supply_net_name*

Specifies the name of the ground supply net to be connected.

-remove

Disconnects the pins of the specified cell instances from the supply net.

-verbose

Prints the pin connection information in details.

DESCRIPTION

This command connects the instance cell's pins to the specified supply nets. The design must be loaded into the session using the **open_mw_cel** command before the **connect_pg_net** command is executed.

EXAMPLES

The following example connects pins to the supply nets VDD and VSS.

```
pr_shell> connect_pg_net -cells FI1 -power_pin vdd -power_net VDD -ground_pin vss -ground_net VSS
1

The following example disconnects pins from the supply net.

pr_shell> connect_pg_net -cells FI1 -power_pin vdd -ground_pin vss -remove
1
```

SEE ALSO

open_mw_cel(2)

connect_supply_net

Connects the supply net to specified supply ports or pins. This command is part of UPF definition of virtual power and ground network.

SYNTAX

```
status connect_supply_net
  -ports list
  supply_net_name
```

Data Types

list	list
supply_net_name	string

ARGUMENTS

-ports *list*

Specifies supply ports to be connected to the supply net. The name is hierarchical name.

supply_net_name

Specifies the name of the supply_net to be connected. The name should be a simple (nonhierarchical) name. The net must exist in the current scope. This argument is required. If you use this option in UPF mode, it specifies which supply ports or instance power ground pins are to be connected with the supply net.

DESCRIPTION

This command provides an explicit connect of a supply net to any supply ports or pins and overrides (has higher precedence than) the auto-connection semantics that might otherwise apply. If a design element is not connected explicitly to any supply net using the **connect_supply_net** command, it shares the primary power/ground supply net with the power domain it belongs to.

The instance that contains the pin must also be in the extend of the same power domain. A supply port cannot connect to two nets in the same domain. Note that the commands does not specify whether the net connects to the inside or outside side of the port. The port side is auto-derived from the net domain and port scope.

Note that explicit connections cannot be made to cell PG pins that are created based on default PG pin names by the tool.

EXAMPLES

The following example connects a VSS supply net with VSS supply port and ground pin of cell INST_1/u1:

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net VSS -domain PD1
VSS
prompt> create_supply_port VSS -domain PD1
VSS
prompt> connect_supply_net VSS -ports VSS
1
prompt> connect_supply_net VSS -ports INST_1/u1/GND
1
```

SEE ALSO

```
create_power_domain(2)
create_supply_net(2)
create_supply_port(2)
get_supply_nets(2)
get_supply_ports(2)
report_supply_net(2)
```

convert_mw_lib

Converts the cell data of a Milkyway library to the latest release version and saves the converted cell data.

SYNTAX

```
status convert_mw_lib
   [-cell_name mw_cel]
   [-all]
   mw_lib
```

Data Types

```
string mw_lib
string mw_cel
```

ARGUMENTS

-cell_name *mw_cel*

Specifies the name of the Milkyway cell to be converted. When the option is specified, all child soft macro cells in the same library are converted together.

-all

Forces conversion of all Milkyway cells under the Milkyway design library.

mw_lib

Specifies the Milkyway library that contains the Milkyway cells to be converted.

DESCRIPTION

This command converts all the Milkyway cells of the specified Milkyway library, except those in the reference libraries. If a Milkyway cell was created by a prior release of the IC Compiler tool, the cell might be in an older format and requires data conversion before it can be opened within the rail analysis session. The converted cells are supported in any tool that is compatible with the current version of the IC Compiler tool. This command saves the converted Milkyway cells back to the specified Milkyway library.

It is recommended that you convert all Milkyway cells by using the **convert_mw_lib** command, if you are going to use the latest release version of the tool. Doing this minimizes runtime and memory consumption for conversion.

If you run the **open_mw_cel** command to open a Milkyway cell of an older format, the **open_mw_cel** command automatically converts the cell and saves the converted cell in memory only. To save the converted data to the

Milkyway library on disk, run the **convert_mw_lib** command first before opening it with the **open_mw_cel** command.

EXAMPLES

The following example converts the *top* Milkyway cell of the Milkyway library *foo_library*:

```
pr_shell> convert_mw_lib foo_library -cell_name top
1
```

SEE ALSO

open_mw_cel(2)

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection copy_collection
   collection1
```

Data Types

collection1	collection

ARGUMENTS

collection1

Specifies the collection to be copied. If an empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is an empty collection).

DESCRIPTION

This command is no longer needed and is provided only to make old scripts work without modification.

EXAMPLES

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different that having multiple references to the same collection, except it is slower.

```
pt_shell> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
pt_shell> set c2 [copy_collection $c1]
{"U1", "U10", "U11", "U12"}
pt_shell> unset c1
pt_shell> query_objects $c2
{"U1", "U10", "U11", "U12"}
```

SEE ALSO

`collections(2)`

copy_mw_cel

Copies Milkyway designs from a source design library to a target design library.

SYNTAX

```
status copy_mw_cel
   -from source_mw_cel_name
   [-to target_mw_cel_name]
   [-from_library source_library_name]
   [-to_library target_library_name]
   [-hierarchy]
   [-overwrite]
```

Data Types

```
source_mw_cel_name      collection
target_mw_cel_name      string
source_library_name     string
target_library_name     string
```

ARGUMENTS

-from *source_mw_cel_name*

Specifies the Milkyway designs to be copied. You can specify one single design or multiple designs when the -hierarchy option is not used. Otherwise, you can specify only one single design.

If you specify a version in the source file name, only that version of the Milkyway design is copied. If you do not specify a version in the source file name, the latest version of the Milkyway design is copied.

-to *target_mw_cel_name*

Specifies the name of the target Milkyway design.

If you specify multiple source Milkyway designs in the -from option, the tool assumes that you want to copy them without changing the names and ignores this option.

The -to and -hierarchy options are mutually exclusive.

If you do not specify this option, the source Milkyway designs are copied to the target library without changing the name.

-from_library *source_library_name*

Specifies the design library that contains the source Milkyway designs.

-to_library *target_library_name*

Specifies the target library to which the Milkyway designs are copied. Generally, you do not need to open the target library before running this command.

If you do not specify this option, the tool uses the current design library as the target library.

-hierarchy

Copies the source Milkyway design and its subdesigns to the target design library.

The -hierarchy and -to are mutually exclusive.

-overwrite

Overwrites the target Milkyway design when copying. This option cannot be used with the -hierarchy option.

DESCRIPTION

This command copies the specified Milkyway designs from the source design library to the target design library. You must have write permission for the target design library. Only read permission is required for the source design library.

You can use the -hierarchy option to copy both the top-level design and its subdesigns. Subdesigns that are instantiated from the reference library stay in the reference library and are not copied to the target design library.

The target library sets the reference path to those reference libraries. The latest version of any open and unsaved Milkyway design in the hierarchical tree is saved from memory into the target design library.

If the open Milkyway design is not the latest version, the Milkyway design is copied from disk instead of memory. If the Milkyway design is not open, the Milkyway design on disk is copied.

The copied Milkyway designs include all information from the original Milkyway design. The copied Milkyway designs have their versions set to 1 if the target library is a new library.

If the target library is not a new library, the version of the copied Milkyway design is incremented by 1 from the latest version before the copy, unless you use the -overwrite option.

EXAMPLES

The following example copies the design named IP_test_TOP and all of its subdesigns from the mwlib_top library to a library named newlib.

```
prompt> copy_mw_cel -hierarchy -from_library mwlib_top -from IP_test__TOP -to_library newlib
Copying Cell: IP_testTOP
    From: pr/mwlib_top
    To: newlib

Copying cell:IP_test_TOP.FILL;1 into library:newlib.
Copying cell:IP_test_TOP.CEL;1 into library:newlib.
Copying Cell: IP_test
    From: pr/mwlib_top
    To: newlib

Copying cell:IP_test.FILL;1 into library:newlib.
Copying cell:IP_test.CEL;1 into library:newlib.
```

SEE ALSO

```
close_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
get_mw_cels(2)
open_mw_cel(2)
remove_mw_cel(2)
rename_mw_cel(2)
mw_support_hier_fill_view(3)
```

copy_mw_lib

Copies a Milkyway library to another location.

SYNTAX

```
status copy_mw_lib
   [-from mw_lib]
   -to lib_name
```

Data Types

```
mw_lib       string
lib_name     string
```

ARGUMENTS

-from *mw_lib*

Specifies the name of the source Milkyway library to be copied. The *mw_lib* value can be a library name or a collection of libraries. By default, the command uses the current Milkyway library.

-to *lib_name*

Specifies the destination Milkyway library name.

DESCRIPTION

The **copy_mw_lib** command copies a Milkyway library to another location. It returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example copies a Milkyway library design to another location:

```
prompt> copy_mw_lib -from design -to design.bak
```

```
1
```

SEE ALSO

rename_mw_lib(2)

cputime

Retrieves the overall user time associated with the current pt_shell process.

SYNTAX

```
float cputime
   [format]
```

Data Types

format	string

ARGUMENTS

format

This argument takes a *print* style formatting string for a floating point number.

DESCRIPTION

This command returns the accumulated user time, in seconds, for the current pt_shell process. If the *format* string is omitted, an integer number of seconds is returned. If the *format* string is specified, a floating point number is returned. The number includes fractions of a second elapsed and is formatted in accordance with given specifier. For detailed information about formatting a floating point argument, see the **print** UNIX man page.

EXAMPLES

```
pt_shell> cputime "%g"
3.74
```

SEE ALSO

mem(2)

create_cell

Creates cells in the current design.

SYNTAX

```
status create_cell
  [-libraries lib_spec]
  [-exact]
  cell_list
  lib_cell
```

Data Types

```
lib_spec        string
cell_list       list
lib_cell        string
```

ARGUMENTS

-libraries *lib_spec*

If this option is specified, PrimeTime resolves the *lib_cell* option from the libraries contained in only the *lib_spec* option. Libraries are searched in the order in which they appear in *lib_spec*. The *lib_spec* value can be a list of library names or collections of libraries loaded into PrimeTime. You can obtain the latter by using the **get_libs** command. You cannot specify this option if a full library cell name has been specified.

-exact

Indicates that names are to be used exactly as specified. Use this option if you want to create a cell that contains a hierarchy or wildcard character as part of the cell name. For more information, see the Naming Cells With Special Characters section.

cell_list

Specifies a list of cells to be created.

lib_cell

Specifies the name of the library cell to which the specified cells are to be linked. The *lib_cell* option can be a library cell object or the name of a library cell. You can obtain the former by using the **get_lib_cells** command. The latter can either be the full library cell name, such as *lib_name/lcell_name*, or the just the base name of the library cell, such as *lcell_name*. You cannot specify the **-libraries** option and explicitly specify the full library cell name. If you invoke PrimeTime with the **-multi_scenario** option, only the library cell base name must be used. For more information, see the Resolved Library Cells section.

DESCRIPTION

This command creates new cells in the current design. Like all other netlist editing commands, for the **create_cell** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **create_cell** command returns a 1 if successful and a 0 if unsuccessful.

Cells are created in scope; that is, at or below the current instance. Cell names are specified as for other commands, using the full hierarchical name relative to the current instance, as in the following example:

```
create_cell u1/u2/u3 class/AN2
```

This command attempts to create a cell named u3 in the hierarchical block u1/u2. The name to the right of the last hierarchical separator is the actual cell name, and the remainder is sent to a *search engine* to find a hierarchical block in which to create the new cell. The operation fails if any of the following are true:

- The search engine cannot find u1/u2.
- The search engine finds multiple blocks that match u1/u2.
- The search engine finds a leaf cell matching u1/u2.

Currently, you cannot create new hierarchy; that is, you cannot create a cell that is an instance of a design. The *lib_cell* must be a leaf library cell.

After creating a cell, you can connect nets to its pins with the **connect_net** command. You can remove cells with the **remove_cell** command.

Resolving Library Cells

If the *lib_cell* has been specified in base name only format, such as without a library from which to resolve it from, PrimeTime resolves the library cell according to the following methodology.

If the **-libraries** option is specified, PrimeTime searches for library cells only in the libraries contained within the *lib_spec*.

Alternatively, if the **-libraries** option is not specified, PrimeTime searches for the library cell in the libraries contained within the **link_path** variable.

The first library cell that is found is used.

Naming Cells With Special Characters

If you want to create a cell whose name contains the current hierarchical separator or wildcard characters used by the search engine, you must do the following:

- Use the **current_instance** command to change the scope to the block in which you want the new cell created.
- Use the **create_cell -exact** command to create the cell.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design,

rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following example, cells are created in the current instance, and at a level below the current instance. Currently, u1 is an instance of a design that has multiple instances. Therefore, it is uniquified as part of the editing operation.

```
pt_shell> create_cell t1 class/AN2
Information: Created cell 't1' in design 'top'. (NED-014)
1

pt_shell> create_cell u1/t1 class/AN2
Uniquifying 'u1' (block1) as 'block1_0'.
Information: Created cell 't1' in 'top/u1'. (NED-014)
1
```

The following example creates cell a/b in the hierarchical block u1/u2. The first attempt fails, because the *exact* option was omitted. Here, u1/u2 is an instance of a design that has multiple instances; therefore, it is uniquified as part of the editing operation. The u1 was already uniquified in the previous example.

```
pt_shell> current_instance u1/u2
u1/u2

pt_shell> create_cell  a/b class/AN2
```

```
Error: Could not create cell 'a/b':
        a not found. (NED-041)
Error: No changes made. (NED-040)
0

pt_shell> create_cell a/b class/AN2  -exact
Uniquifying 'u1/u2' (block2) as 'block2_0'.
Information: Created cell 'a/b' in 'top/u1/u2'. (NED-014)
1
```

In the following example, multiple cells are created using library cells chosen from user-specified libraries using only the library cell base name. The 'lib1' library is searched first as it appears in the *lib_spec* list first and then 'lib2' is searched.

```
pt_shell> create_cell -libraries {lib1 lib2} {u1 u2 u3} ND2
Information: Created cell 'u1' in design 'middle' with 'lib1/ND2'. (NED-014)
Information: Created cell 'u2' in design 'middle' with 'lib1/ND2'. (NED-014)
Information: Created cell 'u3' in design 'middle' with 'lib1/ND2'. (NED-014)
1
```

SEE ALSO

```
connect_net(2)
create_net(2)
current_instance(2)
remove_cell(2)
size_cell(2)
swap_cell(2)
write_changes(2)
link_path(3)
```

create_clock

Creates a clock object.

SYNTAX

```
string create_clock
   -period period_value
   [-name clock_name]
   [-waveform edge_list]
   [-add]
   [-comment comment_string]
   [source_objects]
```

Data Types

period_value	float
clock_name	string
edge_list	list
source_objects	list
comment_string	string

ARGUMENTS

-period *period_value*

Specifies the clock period in library time units. This is the minimum time over which the clock waveform repeats. The *period_value* type must be greater than or equal to zero.

-name *clock_name*

Specifies the name of the clock being created, enclosed in quotation marks. If you do not use this option, the clock gets the same name as the first clock source specified in the *source_objects* option. If you did not specify the *source_objects* option, you must use the *-name* option, which creates a virtual clock not associated with a port or pin. You can use both the *-name* and *source_objects* options to give the clock a more descriptive name than the first source pin or port. If you specify the *-add* option, you must use the *-name* option and the clocks with the same source must have different names.

-waveform *edge_list*

Specifies the rise and fall edge times of the clock waveforms of the clock, in library time units, over an entire clock period. The first time in the *edge_list* is a rising transition, typically the first rising transition after time zero. There must be an even number of edges, and they are assumed to be alternating rise and fall. The edges must be monotonically increasing, except for a special case with two edges, where fall can be less than rise. The numbers should represent one full clock period. If you do not specify this option, a default waveform is assumed, which has a rise edge of 0.0 and a fall edge of *period_value*/2.

-add

Specifies whether to add this clock to the existing clock or to overwrite it. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the *-name* option.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because PrimeTime must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations.

-comment *comment_string*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

source_objects

Specifies the objects used as sources of the clock. The sources can be ports, pins, or nets in the design. If you do not use this option, you must use the *-name* option, which creates a virtual clock not associated with a port, pin, or net. If you specify a clock on a pin that already has a clock, the new clock replaces the old one unless you use the *-add* option. When a net is used as the source, the first driver pin of the net is the actual source used in creating the clock.

DESCRIPTION

Creates a clock object. It is created in the current design and is applied to the specified *source_objects* value. If you do not specify a *source_objects* value, but give a *clock_name* value, a virtual clock is created. You can create a virtual clock to represent an off-chip clock for input or output delay specification. For more information about input and output delay, see the **set_input_delay** and **set_output_delay** man pages.

If one of the *source_objects* values is already the source of a clock, the source is removed from that clock. This clock is eliminated if it has just one source.

The **create_clock** command also defines the waveform for the clock. The clock can have multiple pulses per period.

The **create_clock** command is used together with the **set_clock_latency**, **set_clock_uncertainty**, **set_propagated_clock**, and **set_clock_transition** commands to specify properties of clock networks. By default, a new path group is created for the clock. This new path group brings together the endpoints related to this clock for cost function calculation. To remove the clock from its assigned group, use the **group_path** command to reassign the clock to another group or to the default path group. For more information, see the **group_path** man page.

The new clock has ideal clock latency and transition time; no propagated delay through the clock network is assumed and a transition time of zero is used at the clock source pin. To enable propagated latency for a clock network, use the **set_propagated_clock** command. To set an estimated latency, use the **set_clock_latency** command.

To show information about clocks in the design, use the **report_clock** command. To create a collection of clocks matching a pattern and optionally matching filter criteria, use the **get_clocks** command.

To undo **create_clock**, use the **remove_clock** command.

EXAMPLES

The following example creates a clock on a port named *PHI1* with a period of *10.0*, a rise at *5.0*, and a fall at *9.5*.

```
pt_shell> create_clock PHI1 -period 10 -waveform { 5.0 9.5 }
```

The following example shows a clock named *PHI2* with a falling edge at *5* and a rising edge at *10* with a period of *10*. Because the edges for the *-waveform* option should be ordered as first rise, then fall, and should increase in value, the fall edge can be given as *15*. This places the next falling edge after the first rise edge at *10*.

```
pt_shell> create_clock PHI2 -period 10 -waveform { 10 15 }
```

The following example creates a virtual clock *PHI2* with a period of *10.0*, a rise at *0.0*, and a fall at *5.0*.

```
pt_shell> create_clock -name PHI2 -period 10 -waveform {0.0 5.0}
```

The following example creates a clock named *clk2* with multiple sources.

```
pt_shell> create_clock -name clk2 -period 10 \
-waveform {2.0 4.0} {clkgen1/Z clkgen2/Z clkgen3/Z}
```

The following example creates a clock named *CLK* on pin *u13/Z* with a period of *25*, a fall at *0.0*, a rise at *5.0*, a fall at *10.0*, a rise at *15.0*, and so on.

```
pt_shell> create_clock u13/Z -name CLK -period 25 -waveform { 5 10 15 25}
```

SEE ALSO

```
all_clocks(2)
get_clocks(2)
group_path(2)
remove_clock(2)
report_clock(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_input_delay(2)
set_output_delay(2)
set_propagated_clock(2)
set_clock_transition(2)
```

create_command_group

Creates a new command group.

SYNTAX

```
string create_command_group [-info info_text]
group_name
```

ARGUMENTS

-info *info_text*

Help string for the group

group_name

Specifies the name of the new group.

DESCRIPTION

The **create_command_group** command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the **define_proc_attributes** command, you can move the procedure into the group you created.

The *group_name* can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, **create_command_group** quietly ignores the command. The result of **create_command_group** is always an empty string.

EXAMPLES

The following example demonstrates the use of the **create_command_group** command:

```
prompt> create_command_group {My Procedures} -info "Useful utilities"

prompt> proc plus {a b} { return [expr $a + $b] }
```

```
prompt> define_proc_attributes plus -command_group "My Procedures"

prompt> help
My Procedures:
  plus

 ...
```

SEE ALSO

define_proc_attributes(2)
help(2)
proc(2)

create_generated_clock

Creates a generated clock object.

SYNTAX

```
string create_generated_clock
   [-name clock_name]
   -source master_pin
   [-divide_by divide_factor | -multiply_by multiply_factor |
    -edges edge_list ]
   [-combinational]
   [-duty_cycle percent]
   [-invert]
   [-preinvert]
   [-edge_shift edge_shift_list]
   [-add]
   [-master_clock clock]
   [-pll_output output_pin]
   [-pll_feedback feedback_pin]
   [-comment comment_string]
   source_objects
```

Data Types

clock_name	string
master_pin	list
divide_factor	int
multiply_factor	int
edge_list	list
percent	float
edge_shift_list	list
clock	string
output_pin	list
feedback_pin	list
comment_string	string
source_objects	list

ARGUMENTS

-name *clock_name*

Specifies the name of the generated clock. If you do not use this option, the clock receives the same name as the first clock source specified in the **-source** option. If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-source *master_pin*

Specifies the master clock source (a clock source pin in the design) from which the clock waveform is to be

derived. Note that the actual delays (latency) for the generated clock are computed using its own source pins and not the *master_pin* option.

-divide_by *divide_factor*

Specifies the frequency division factor. For example, **-divide_by 2** specifies that the generated clock period is twice as long as the master clock period.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. For example, **-multiply_by 3** specifies that the generated clock period is one-third of the master clock period.

-edges *edge_list*

Specifies a list of integers that represents edges from the source clock that are to form the edges of the generated clock. The edges are interpreted as alternating rising and falling edges and each edge must be not less than its previous edge. The number of edges must be an odd number and not less than 3 to make one full clock cycle of the generated clock waveform. For example, *1* represents the first source edge, *2* represents the second source edge, and so on.

-combinational

The source latency paths for this type of generated clock only includes the logic where the master clock propagates. The source latency paths do not flow through sequential element clock pins, transparent latch data pins, or source pins of other generated clocks.

-duty_cycle *percent*

Specifies the duty cycle, in percentage, if frequency multiplication is used. Duty cycle is the high pulse width.

-invert

Inverts the generated clock signal (in the case of frequency multiplication and division). This option first creates the generated clock and then inverts the generated clock signal.

-preinvert

Creates a generated clock based on the inverted clock signal. This option first inverts the signal and then creates the generated clock signal.

-edge_shift *edge_shift_list*

Specifies a list of floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified. The values can be positive or negative; positive indicating a shift later in time, while negative indicates a shift earlier in time. For example, *1* indicates that the corresponding edge is to be shifted by one library time unit.

-add

Specifies whether to add this clock to the existing clock or to overwrite it. Use this option to capture the case where multiple generated clocks must be specified on the same source, because multiple clocks fan into the master pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. If you specify this option, you must also use the **-name** and **-master_clock** options.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because PrimeTime must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations.

-master_clock *clock*

Specifies the master clock to be used for this generated clock if multiple clocks fan into the master pin. If you specify this option, you must also use the **-add** option.

-pll_output ***output_pin***

Specifies the output pin of the PLL which is connected to the feedback pin. For single output PLLs, this pin is same as the pin on which the generated clock is defined.

-pll_feedback ***feedback_pin***

Specifies the feedback pin of the PLL. There should be a path in the circuit connecting one of the outputs of the PLL to this feedback pin.

-comment ***comment_string***

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

source_objects

Specifies a list of ports, pins or nets defined as generated clock source objects. When a net is used as the source, the first driver pin of the net is the actual source used in creating the generated clock.

DESCRIPTION

Creates a generated clock object in the current design and also defines a list of objects as generated clock sources in the current design. You can specify a pin or a port as a generated clock object. The command also specifies the clock source from which it is generated. The advantage of using this command is that whenever the master clock changes, the generated clock automatically changes.

The generated clock can be created as one of the following:

- A frequency divided clock by using the **-divide_by** option
- A frequency multiplied clock by using the **-multiply_by** option
- A special divide by one by using the **-combinational** option
- An edge-derived clock by using the **-edges** option

The frequency-divided or frequency-multiplied clock can be inverted by using the **-invert** option. The shifting of edges of the edge-derived clock is specified by using the **-edge_shift** option. The **-edge_shift** option is used for intentional edge shifts and not for clock latency.

The number of edges specified by **-edges** to make one period of the generated clock waveform must be an odd number equal to or greater than 3. For example, in the following command edge 1 indicates the first rising of the generated clock, edge 3 indicates the first falling of the generated clock, and edge 5 indicates the next rising of the generated clock. Note that the period of the generated clock is determined by the final entry in the edge list:

```
create_generated_clock -source clk -edges { 1 3 5 } [get_pins flop/Q]
```

Non-increasing edges as **-edges { 1 1 3 }** are allowed and usually used with the **-edge_shift** option to produce a generated clock pulse independent of the duty cycle of the master clock itself.

If a generated clock is specified with a *divide_factor* value that is a power of 2 (1, 2, 4, ...), the rising edges of the master clock are used to determine the edges of the generated clock. If the *divide_factor* value is not a power of two, the edges are scaled from the master clock edges.

Using the **create_generated_clock** command on an existing *generated_clock* object overwrites its attributes. The **generated_clock** objects are expanded to real clocks at the time of analysis.

The **set_clock_latency**, **set_clock_uncertainty**, **set_propagated_clock**, and **set_clock_transition** commands can reference the generated clock.

For internally generated clocks, PrimeTime automatically computes the clock source latency if the master clock of the generated clock has propagated latency and no user-specified value for generated clock source latency exists. If the master clock is ideal and has source latency and there is no user-specified value for the generated clock's source latency, zero source latency is assumed.

To display information about generated clocks, use the **report_clock** command.

EXAMPLES

The following example creates a frequency divide-by 2 generated clock.

```
pt_shell> create_generated_clock -divide_by 2 \
          -source [get_pins CLK] [get_pins pinf]
```

The following example creates a frequency divide-by 3 generated clock. If the master clock period is 30, and master waveform is {24 36}, the generated clock period is 90 with waveform {72 108}.

```
pt_shell> create_generated_clock -divide_by 3 \
          -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a frequency multiply-by 2 generated-clock with a duty cycle of 60%.

```
pt_shell> create_generated_clock -multiply_by 2 -duty_cycle 60 \
          -source [get_pins CLK] [get_pins pinf1]
```

The following example creates a frequency multiply-by 3 generated-clock with a duty cycle equal to the master clock duty cycle. If the master clock period is 30, and master waveform is {24 36}, the generated clock period will be 10 with waveform {8 12}.

```
pt_shell> create_generated_clock -multiply_by 3 \
          -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a generated clock with edges at first, third, and fifth edges of the master clock source.

```
pt_shell> create_generated_clock -edges {1 3 5} \
          -source [get_pins CLK] [get_pins pinf2]
```

The following example shows the generated clock in the previous example with each derived edge shifted by one time unit.

```
pt_shell> create_generated_clock -edges {1 3 5} -edge_shift {1 1 1} \
          -source [get_pins CLK] [get_pins pinf2]
```

The following example creates an inverted clock.

```
pt_shell> create_generated_clock -divide_by 1 -invert
```

The following example creates a rising edge pulse triggered by the rising edge of its master clock.

```
pt_shell> create_generated_clock -edges {1 1 3} -edge_shift {0 5 0}  \
          -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a falling edge pulse triggered by the rising edge of its master clock with a period 10.

```
pt_shell> create_generated_clock -edges {1 1 3} -edge_shift {0 5 0} \
          -invert -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a frequency doubling pulse. A rising edge pulse triggered by both rising and falling edges of its master clock with a period of 10.

```
pt_shell> create_generated_clock -edges {1 1 2 2 3} -edge_shift {0 2.5 0 2.5 0} \
          -source [get_pins CLK] [get_pins pinf2]
```

SEE ALSO

check_timing(2)
create_clock(2)
get_generated_clocks(2)
remove_generated_clock(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_false_path(2)
set_propagated_clock(2)

create_mtcmos_switch_cell_sequence

Creates an event sequence to be used in the rush current analysis.

SYNTAX

```
status create_mtcmos_switch_cell_sequence
   [-import file_name]
   [-from_timing_analysis]
   [-offset value]
   [-switch_pin_group_number group_number]
```

Data Types

```
string file_name
```

ARGUMENTS

-import *file_name*

Specifies a file that is a PrimeRail scheme based event sequence script file. The syntax of this script file is shown as follows.

;definePortInstancePMCellEvent (geGetEditCell) "inst_name" "pg_port_name" num_port '(p1 p2 ...) num_event '(t0 s0 t1 s1 ...)> ;definePMCellPortInstTransition (geGetEditCell) "inst_name" "signal_port_name" riseSlew fallSlew>

;;; Unit Info ; time unit = 1e-09 seconds

"definePortInstancePMCellEvent" is the scheme command name used to specify a mtcmos cell event. The "inst_name" argument after "(geGetEditCell)" is the name of the mtcmos cell instance, whose event is specified afterwards. The "pg_port_name" argument is the pg pin name of the mtcmos cell instance, which connects to a main supply net. The num_port argument defines the number of the control pins on the mtcmos cell instance. The '(p1 p2 …) argument defines the list of control pin names which are referred to in the event state information . The number of control pin names,i.e., p1, p2,…, must be equal to the number defined in the num_port argument's value. The num_event argument defines the number of events. The '(t0 s0 t1 s1…) argument defines the sequence of events as a list of time and control pin state pairs. The first event has a state s0 ioccuring at time t0, and the second event has a state s1 occuringat time t1. The s0 or any state argument is a string of characters whose length is the number of control pin names. Each of these characters can have one of these letters 'x', '1', '0', 'f' or 'r' which indicate unknown, 1 logic level, 0 logic level, falling transition or rising transition control pin states respectively. The character index in the state string refers to the state of the corresponding indexed control pin list item. The number of events specified in this sequence argument must be equal to the number defined by the num_event value.

In the script file, lines starting with ";" or "#" are treated as comments. The only exception to this assumption is with the time unit syntax discussed below, which will be used for scaling the time values used in the file. If there is no time unit specified, the time unit of seconds will be used.

The following script is an example of the PrimeRail scheme based event sequence script.

;;; Unit Info ; time unit = 1e-09 seconds ; power unit = 1 w

definePortInstancePMCellEvent (geGetEditCell) "header0" "VDDT" 2 '(EN CNT) 2 '(9.702 f1 9.886 0f)
definePortInstancePMCellEvent (geGetEditCell) "header1" "VDDT" 2 '(EN CNT) 2 '(9.755 r0 9.985 1r)
definePortInstancePMCellEvent (geGetEditCell) "header2" "VDDT" 2 '(EN CNT) 2 '(19.12 1f 20.66 f0)

In the above example, the time unit used in the script is 1e-9 second, which is specified in second comment line. There are 3 event sequences specified in the script, one for each mtcmos switch cell. The first event sequence (of only one event) is defined in the first uncommented line, beginning with "definePortInstancePMCellEvent". The mtcmos cell instance of this first event sequence is named "header0", whose pg pin "VDDT" connects to the main power supply net. There are two control pins, whose names are "EN" and "CNT". There are two events in the event sequence, which occurs at 9.702e-09 second and 9.886e-09 second. For the first event, the pin state of "EN" is falling, and the state of pin "CNT" is logic "1". For the second event, the pin state of "EN" is logic "0", and the state of pin "CNT" is falling.

Specifies a file that is a PrimeRail Gen1 event sequence script file.

-from_timing_analysis

Enables the use of a power-up sequence that is derived from STA. The arrival times and slews of the switch cells control pins will be used in the derivation of such sequence. This is useful if the user doesn't have a VCD or a power-up sequence file,

-offset *value*

Applies an offset to all event start times for the specified group of switch cells pins. If no group is specified, then the offset is applied to the pins of all the switch cells in the design.

-switch_pin_group_number *group_number*

Specifies the group of switch cell pins to which the offset is applied. This option can only be used in conjunction with the -offset option.

DESCRIPTION

The **create_mtcmos_switch_cell_sequence** command creates an event sequence for mtcmos switch cell. This event sequences created here have higher priority than the event sequences extracted from a VCD file and are therefore used in the subsequent rush current analysis invocations when both are available in the current user session. Note that repeated use of the command with any of the above mutually exlusive options will over-write the previous events sequence.

EXAMPLES

This example shows how to create event sequences by importing a mtcmos switch cell sequence script file.

```
pr_shell> create_mtcmos_switch_cell_sequence -import sequence_file
```

This example shows how to create event sequences based on STA.

```
pr_shell> create_mtcmos_switch_cell_sequence -from_timing_analysis
```

```
Information: Computed a power-up sequence for the design's switch cells based on STA. Refer to file I
```

The following two examples shows how to create event sequences by importing a mtcmos switch cell sequence script file.

```
pr_shell> create_mtcmos_switch_cell_sequence -offset 2e-9

The following offsets will be applied to the switch cells event times.
Switch Pin Group         Offset
0                        2.000000e-09
1                        2.000000e-09

pr_shell> create_mtcmos_switch_cell_sequence -offset +2e-9 -switch_pin_group_number 0

The following offsets will be applied to the switch cells event times.
Switch Pin Group         Offset
0                        2.000000e-09
1                        0.000000e+00
```

SEE ALSO

```
remove_mtcmos_switch_cell_sequence(2)
report_mtcmos_switch_cell_sequence(2)
calculate_rush_currents(2)
```

create_mw_cel

Creates a Milkyway design.

SYNTAX

```
status create_mw_cel
   [-view CEL | FRAM | FILL | err | ILM]
   [-verbose]
   mw_cel_name
```

Data Types

```
mw_cel_name     string
```

ARGUMENTS

-view CEL | FRAM | FILL | err | ILM

Specifies the view name of the Milkyway design to create. The **CEL**, **FRAM**, **FILL**, **err** and **ILM** arguments are mutually exclusive; you can use only one. By default, the command creates a **CEL** view Milkyway design.

-verbose

Prints additional messages.

mw_cel_name

Specifies the name of the Milkyway design to create. The length of a Milkyway design name must be less than 127 characters. Do not specify a view name or version in the *mw_cel name*. For example, *test.CEL* or *test.CEL;2* are not valid values for *mw_cel name*. To specify the view, use *-view* option.

DESCRIPTION

This command creates a new Milkyway design. It automatically opens this newly-created Milkyway design and sets it as the current Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a Milkyway design named *top* for CEL view.

```
prompt> create_mw_cel top
{"top"}


prompt> current_mw_cel
{"top"}
```

The following example creates a Milkyway design named *top* for err view.

```
prompt> create_mw_cel top -view err
{"top"}


prompt> current_mw_cel
{"top"}
```

SEE ALSO

```
close_mw_cel(2)
copy_mw_cel(2)
current_mw_cel(2)
get_mw_cels(2)
open_mw_cel(2)
remove_mw_cel(2)
rename_mw_cel(2)
save_mw_cel(2)
mw_cel_collection(2)
```

create_mw_lib

Creates a Milkyway library.

SYNTAX

```
status create_mw_lib
   [-technology technology_file_name]
   [-plib plib_file_name]
   [-bus_naming_style style]
   [-mw_reference_library lib_list]
   [-reference_control_file rc_file_name]
   [-open]
   libName
```

Data Types

technology_file_name	string
plib_file_name	string
style	string
lib_list	string
rc_file_name	string
libName	string

ARGUMENTS

-technology *technology_file_name*

Specifies the name of the technology file for the newly created Milkyway library. This option and **-plib** are mutually exclusive.

-plib *plib_file_name*

Specifies the name of the .plib file for the newly created Milkyway library. Valid file is of .plib format or .pdb format. This option and **-technology** are mutually exclusive. If routing directions in the .plib file are used to set preferred routing directions for the design library, you must specify the reference libraries by using the **-reference_control_file** or **-mw_reference_library** option.

-bus_naming_style *style*

Specifies the bus naming style for the library. The default bus naming style is [%d].

-mw_reference_library *lib_list*

Specifies a list of reference libraries to be used for the new library.

-reference_control_file *rc_file_name*

Specifies the reference control file used to set reference library information for the new library.

-open

Opens the library after creation.

libName

Specifies the name of the Milkyway library to be created.

DESCRIPTION

This command creates a Milkyway library.

The **-technology** and **-plib** options are mutually exclusive, and at least one of them must be specified.

Some advanced technologies require more than the default 255 layers. In these cases, you can extend the number of layers to 4095 by executing the **extend_mw_layers** command before you exeucte the **create_mw_lib** command. Once created in the default 255-layer mode or extended 4095-layer mode, the Milkyway library must remain in that layer mode. For details, see the man page for the **extend_mw_layers** command.

By default, the newly created Milkyway library is not open in the current session. To manipulate it, first run the **open_mw_lib** command. The library can also be opened by using the **-open** option, but to makes the scripts more reusable, use **open_mw_lib**.

All strings stored in this library, as well as in designs belonging to it, are case-sensitive, and all string operations are performed as case-sensitive.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a Milkyway design library with a technology file and bus naming style as [%d].

```
prompt> create_mw_lib design -technology test.tf \
        -bus_naming_style {[%d]}
1
```

SEE ALSO

```
close_mw_lib(2)
create_lib_track(2)
extend_mw_layers(2)
open_mw_lib(2)
```

create_net

Creates nets in the current design.

SYNTAX

```
status create_net
   [-exact]
   net_list
```

Data Types

```
net_list     list
```

ARGUMENTS

-exact

Indicates that names are to be used exactly as specified. Use this option if you want to create a net that contains a hierarchy or wildcard character as part of the net name. For more information, see "Naming Nets with Special Characters".

net_list

Specifies a list of nets to be created.

DESCRIPTION

The **create_net** command creates new nets in the current design. Like all other netlist editing commands, for the **create_net** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **create_net** command returns a 1 if successful and a 0 if unsuccessful.

The **create_net** command behaves almost identically to the **create_cell** command. Nets are created in scope; that is, at or below the current instance. Net names are specified as for other commands, using the full hierarchical name relative to the current instance, as in the following example:

```
create_net u1/u2/new_net class/AN2
```

This command attempts to create a net named new_net in the hierarchical block u1/u2. The name to the right of the last hierarchical separator is the actual net name, and the remainder is sent to the search engine to find a hierarchical block in which to create the new net. The operation fails if any of the following are true:

- The search engine cannot find "u1/u2".

- The search engine finds multiple blocks that match "u1/u2".
- The search engine finds a leaf cell matching "u1/u2".
- The search engine finds a net, port or hierarchical pin matching "u1/u2/new_net".

Naming Nets With Special Characters

If you want to create a net whose name contains the current hierarchical separator or wildcard characters used by the search engine, you must do the following:

- Use **current_instance** to change the scope to the block in which you want the new net created.
- Use **create_net -exact** to create the net.

For examples, see the EXAMPLES section.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

The following example creates nets in the current instance, and at a level below the current instance. Currently, u1 is an instance of a design that has multiple instances. Therefore, it is uniquified as part of the editing operation.

```
pt_shell> create_net new_net1
Information: Created net 'new_net1' in design 'top'. (NED-016)
1

pt_shell> create_net u1/new_net1
Uniquifying 'u1' (block1) as 'block1_0'.
Information: Created net 'new_net1' in 'top/u1'. (NED-016)
1
```

The following example creates net a/b in the hierarchical block u1/u2. The first attempt fails, because the **-exact** option was omitted. Here, u1/u2 is an instance of a design that has multiple instances; therefore, it is uniquified as part of the editing operation. u1 was already uniquified in the previous example.

```
pt_shell> current_instance u1/u2
u1/u2

pt_shell> create_net a/b
Error: Could not create net 'a/b':
        a not found. (NED-041)
Error: No changes made. (NED-040)
0

pt_shell> create_net a/b -exact
Uniquifying 'u1/u2' (block2) as 'block2_0'.
Information: Created net 'a/b' in 'top/u1/u2'. (NED-016)
1
```

SEE ALSO

```
create_cell(2)
remove_net(2)
size_cell(2)
swap_cell(2)
write_changes(2)
```

create_operating_conditions

Creates a new set of operating conditions in a library.

SYNTAX

```
int create_operating_conditions
   -name name
   -library library_name
   -process process_value
   -temperature temperature_value
   -voltage voltage_value
   [-tree_type tree_type]
   [-calc_mode calc_mode]
   [-rail_voltages rail_value_pairs]
```

Data Types

name	string
library_name	string
process_value	float
temperature_value	float
voltage_value	float
tree_type	string
calc_mode	string
rail_value_pairs	list

ARGUMENTS

-name *name*

Specifies the name of the new set of operating conditions.

-library *library_name*

Specifies the name of the library for the new operating conditions.

-process *process_value*

Specifies the process scaling factor for the operating conditions. Allowed values are 0.0 through 100.0.

-temperature *temperature_value*

Specifies the temperature value, in degrees Celsius, for the operating conditions. Allowed values are -300.0 through +500.0.

-voltage *voltage_value*

Specifies the voltage value, in volts, for the operating conditions. Allowed values are 0.0 through 1000.0.

-tree_type *tree_type*

Specifies the tree type for the operating conditions. Allowed values are *best_case_tree*, *balanced_tree* (the default), or *worst_case_tree*. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

-calc_mode *calc_mode*

For use only with DPCM libraries. Specifies the DPCM delay calculator mode for the operating conditions; analogous to the *-process* option used in Synopsys libraries. Allowed values are *unknown* (the default), *best_case*, *nominal*, or *worst_case*. The default behavior (*unknown*) is to use worst case values during analysis similarly to worst_case. If **-rail_voltages** are specified, the command sets all (worst_case, nominal, and best_case) voltage values.

-rail_voltages *rail_value_pairs*

Specifies a list of name-value pairs that defines the voltage for each specified rail. The name is one of the rail names defined in the library; the value is the voltage to be assigned to that rail. By default, rail voltages are as defined in the library; use this option to override the default voltages for specified rails.

DESCRIPTION

This command creates a new set of operating conditions in the specified library. A technology library contains a fixed set of operating conditions; this command allows you to create new, additional operating conditions.

To see the operating conditions defined for a library, use the **report_lib** command.

To set operating conditions on the current design, use the **set_operating_conditions** command.

EXAMPLES

The following example creates a new set of operating conditions called WC_CUSTOM in the library "tech_lib", specifying new values for process, temperature, voltage, and tree type. By default, rail voltages remain as defined in the library.

```
pt_shell> create_operating_conditions -name WC_CUSTOM \
-library tech_lib -process 1.2 -temperature 30.0 -voltage 2.8 \
-tree_type worst_case_tree
```

The following example creates a new set of operating conditions called OC3, in the library "IBM_CMOS5S6_SC", specifying new values for process, temperature, voltage, and rail voltages for rails VTT and VDDQ. By default, the tree type *balanced_tree* is used.

```
pt_shell> create_operating_conditions -name OC3 \
-lib IBM_CMOS5S6_SC -proc 1.0 -temp 100.0 -volt 4.0 \
-rail_voltages {VTT 3.5 VDDQ 3.5}
```

SEE ALSO

```
set_operating_conditions(2)
report_lib(2)
```

create_pg_pin_only_lib_cell

Creates library cells for PG-pin-only cell masters.

SYNTAX

```
status create_pg_pin_only_lib_cell
   cell_master_name_list
   -power_pins power_pin_list
   -ground_pins ground_pin_list
   -bias_pins bias_pin_list
```

Data Types

```
list cell_master_name_list
list power_pin_list
list ground_pin_list
list bias_pin_list
```

ARGUMENTS

cell_master_name_list

Specifies a list of cell masters for which the command creates a library cell.

-power_pins *pin_name:type voltage*

Specifies a list of power pins. Supported pg pin types are primary, backup or internal. Default type is primary.

-ground_pins *pin_name:type voltage*

Specifies a list of ground pins. Supported pg pin types are primary, backup or internal. Default type is primary.

-bias_pins *pin_name:type voltage*

Specifies a list of bias pins. Supported bias pin types are nwell, pwell, deep_nwell or deep_pwell. User must specify bias pin type.

DESCRIPTION

This command creates an in-memory library cell for a PG-pin-only cell master which does not exist in the library. Execute the command before the **open_mw_cel**/**open_block** command. When creation process is complete, the command saves the created library cell in the rail_sidefile_lib_0 library.

To examine the created library cell, use the **report_pg_pin_only_lib_cell** command.

EXAMPLES

Example 1: The example creates a PG-pin-only cell reference, called DECAP1, for the cell master DECAP1, and reports the content of the created library cell DECAP1. Then run the **open_mw_cel**, **get_cell** and **get_lib_cell** commands to see if the PG-pin-only cell is connected to VDD.

```
pr_shell> create_pg_pin_only_lib_cell DECAP1 \
            -power_pins {VDD 1.0} -ground_pins {VSS 0.0}
1
pr_shell> report_pg_pin_only_lib_cell DECAP1

****************************************
Report : PG-pin-only library cells
Design :
****************************************

  Lib cell              Power pins          Ground pins         Bias pins
--------------------------------------------------------------------------------
  DECAP1                VDD:p (1.0V)        VSS:p (0V)
1
pr_shell> open_mw_cel -lib design cell
1
pr_shell> get_cell -of [get_supply_net VDD]
{...inst1, inst2, ..., decap1, decap2,...}
pr_shell> get_lib_cell -of [get_cell -of [get_supply_net VDD]]
{...INV1, ..., DECAP1,...}
```

Example 2: Continuing from Example 1, the following example retrieves a collection of PG pins for the PG-pin-only cells.

```
pr_shell> open_mw_cel -lib design cell
1
pr_shell> get_pg_pins -of [get_cell -of [get_supply_net VDD]]
{...decap1/VDD, decap1/VSS, decap2/VDD, decap2/VSS, ...}
1
```

SEE ALSO

report_pg_pin_only_lib_cell(2)

create_placement_blockage

Creates a new placement blockage.

SYNTAX

```
status create_placement_blockage
   -bbox coordinate_list
   [-name blockage_name]
```

Data Types

```
coordinate_list   list
blockage_name     string
```

ARGUMENTS

-bbox *coordinate_list*

Specifies the lower-left and upper-right coordinates (in microns) of the bounding box of the placement blockage in the following format: {*lower_left_x lower_left_y upper_right_x upper_right_y*}.

-name *blockage_name*

Specifies the optional name of the blockage.

DESCRIPTION

This command creates a new placement blockage that controls the placement of ECO cells when you use the **fix_eco_drc** or **fix_eco_timing** command with the **-physical_mode** option. PrimeTime does not resize cells or insert buffers within placement blockages.

You can create a placement blockage only after PrimeTime loads the physical database. To specify the file that contains **create_placement_blockage** commands, use the **set_eco_options** command with the **-physical_constraint_file** option. When you use this option, PrimeTime loads the physical database, reads the constraint file, and creates blockage areas before fixing violations.

EXAMPLES

The following example creates a placement blockage named placeblockage_10.

```
pt_shell> create_placement_blockage -bbox {0 0 100 100} -name placeblockage_10
```

SEE ALSO

create_voltage_area(2)
fix_eco_drc(2)
fix_eco_timing(2)
set_eco_options(2)
write_changes(2)

create_power_rail_mapping

Maps the power rails defined in the libraries to the physical power rails existing in the design.

SYNTAX

```
status create_power_rail_mapping
   [-lib_rail_name power_rail_name]
   [-cells cell_list]
   [-off_condition condition]
   [-default]
   design_rail
```

Data Types

```
power_rail_name        string
cell_list              list
condition              string
design_rail            string
```

ARGUMENTS

-lib_rail_name *power_rail_name*

Specifies the power rail defined in the library. It can take the power rail name defined in the library and also take "all". The power rail name can be the name of the **voltage_map** command defined in the library. By using the **-lib_rail_name all** option, the mapping is applied to all the power rails in each library associated with the library cells. If the **-lib_rail_name** option is not used, the mapping is applied to the default library power rail in each library associated with the library cells.

-cells *cell_list*

Specifies the instance names or collections of instances. The instance can be hierarchical or leaf instances. For an instance block, only the top instance needs to be specified. All the instances inside such block will be affected by the mapping. Without this option, the mapping will be created for the whole design.

-off_condition *condition*

Specifies the power-off condition for the design power rail. At power-off state, the portion of design that is powered off by the specific design power rail will not contribute any dynamic or static power. The Boolean expression takes net names plus operators. The operators supported are: (,), !, +, *, &, |, ^, '. When the expression is evaluated to be "1", the design power rail is in power-off state. If the expression is evaluated to be "0" or any unknown state, the design power rail is in power-on state. Ensure that the states of the nets used in the Boolean expression are clearly defined to avoid any unexpected results. If the **-power_off** option is omitted, the design power rail is assumed to be on all the time.

When switching activity information is specified by the VCD file, the state of the power control signal nets are monitored. The power-off expression is evaluated if there is any state change on these control nets. When a

specific power rail is powered-off, neither dynamic nor static power will be dissipated on the cell (or part of cell) powered by this rail.

When switching activity information is specified by statistical toggle rate and state probability, the state probability of the power rail being on is calculated. As a default, only leakage power is scaled by the state probability of the associated power rail being on. Set the **power_scale_dynamic_power_at_power_off** variable to true, if dynamic power needs to be scaled too. Dynamic power scaling is only needed if the statistical switching activity includes toggles happened when the rail is being powered off.

-default

Specifies the design power rail to be the default rail in the design. If not specified, the design power rail defined by the first **create_power_rail_mapping** command is assigned as the default design rail.

design_rail

Defines the power rail used in the design.

DESCRIPTION

Use this command to map the power rails defined in the libraries to the physical power rails existing in the design. Such rail mapping is done at cell (instance) level, which gives you the flexibility to connect the same library cell to different physical power rails in a design. The design power rails specified in this command are virtual power rails, which means that the rail connections do not actually exist in the netlist. No voltage level violation will be checked for such virtual rail connections. It is allowed to connect cells with different voltages to the same virtual design power rail. This command enables PrimeTime PX to create collections of instances on different power rails. Thus, provides the capability and flexibility to create rail-based power reporting at design level. This command also enables PrimeTime PX to be sensitive to the power-on/off control signals associated with the design power rails.

Single-rail or multi-rail cell in PrimeTime PX is based on its cell definition in logic library. A multi-rail cell can have rail specific internal or leakage power tables defined in the library. So that power consumption can be reported for different rails.

This command can create the rail mapping for the whole design or for specific instances (hierarchical or leaf-level). Multiple **create_power_rail_mapping** commands can be issued, the latter overwrites the previous settings if it overlaps.

For single-rail design, if not specified, the default library rail will be mapped to the default design rail internally.

The **report_power_rail_mapping** command can be used to report existing power rail mapping information.

EXAMPLES

The following example shows how to create rail mapping and generate rail-based power reports for a multi-rail design. There are three instance blocks in this design: block1, block2 and ls_block. Block block1 and block2 are both single-rail instance blocks with all the leaf cells powered by the default library power rail defined in each associated logic library. Block block1 is connected to design power rail VDD1 in the design while block block2 is connected to design power rail VDD2 in the design. Block ls_block is a dual-rail instance block with rail V1 and V2 defined in its cells' logic library. Library power rail V1 is connected to design power rail VDD1 and library power rail V2 is connected to design power rail VDD2. The **current_power_rail** command can select the rail of interest and the **update_power** command can generate the power analysis results just for the interested rails. As a

result, report Power_VDD1.rpt file will only contain the power consumed on design power rail VDD1 and report Power_VDD2.rpt will only contain the power consumed on design power rail VDD2. Since the **current_power_rail all** command will mark all the design power rails to be interested, report Power_all.rpt file will contain the total power consumed for entire design.

```
pt_shell> create_power_rail_mapping VDD1 -cells block1
pt_shell> create_power_rail_mapping VDD2 -cells block2
pt_shell> create_power_rail_mapping VDD1 -lib_rail_name V1 -cells ls_block
pt_shell> create_power_rail_mapping VDD2 -lib_rail_name V2 -cells ls_block
pt_shell> current_power_rail VDD1
pt_shell> update_power
pt_shell> report_power > Power_VDD1
pt_shell> current_power_rail VDD2
pt_shell> update_power
pt_shell> report_power > Power_VDD2
pt_shell> current_power_rail all
pt_shell> update_power
pt_shell> report_power > Power_all
```

SEE ALSO

current_power_rail(2)
read_saif(2)
read_vcd(2)
report_power(2)
report_power_rail_mapping(2)
set_switching_activity(2)
update_power(2)
power_scale_dynamic_power_at_power_off(3)

create_power_switch

Creates a power switch at the specified power domain. This command is supported only in the UPF mode.

SYNTAX

```
status create_power_switch
   switch_name
   -domain domain_name
   -output_supply_port {port_name supply_net_name}
   -input_supply_port  {port_name supply_net_name}
   -control_port {port_name net_name}
   -on_state {state_name input_supply_port {boolean_function}}
   [-off_state {state_name {boolean_function}}]
   [-on_partial_state {state_name input_supply_port {boolean_function}}]
   [-error_state {state_name {boolean_function}}]
   [-ack_port {port_name net_name [{boolean_function}]}]
   [-ack_delay {port_name delay}]
   [-supply {port_name supply_net_name}]
```

Data Types

```
switch_name          string
domain_name          string
port_name            string
supply_net_name      string
net_name             string
state_name           string
input_supply_port    string
boolean_function     list
delay                string
supply_net_name                string
```

ARGUMENTS

switch_name

Specifies the power switch to create. The name should be a simple (nonhierarchical) name. You cannot create a power switch with the same name as the existing power switch in a specified power domain. This option is required.

-domain *domain_name*

Specifies the power domain that contains the power switch. If the power domain with the specified name does not exist in the current scope, the command fails. This option is required.

-output_supply_port {*port_name supply_net_name*}

Specifies the output port of the power switch and the supply net where the port connects. The command fails when,

- A port exists with the same name on the power switch.
- A supply net with the same name does not exist in the current scope. This option is required.

-input_supply_port {*port_name supply_net_name*}

Specifies the input port of the power switch and the supply net where the port connects. The command fails when,

- A port exists with the same name on the power switch.
- A supply net with the same name does not exist in the current scope. This option is required.

-control_port {*port_name net_name*}

Specifies the control port of the power switch and the logical net where this port connects. The command fails when,

- A port with the specified name already exists on the power switch.
- A net with the specified name does not exist. This option is required. You can specify this option more than one time. One power switch can have multiple control ports.

-on_state {*state_name input_supply_port* {*boolean_function*}}

Specifies the on state, the relevant input supply port, and its Boolean function. You can specify this option multiple times.

-off_state {*state_name* {*boolean_function*}}

Specifies the off state and its relevant Boolean function. This option is currently ignored by the PrimeTime tool.

-on_partial_state {*state_name input_supply_port* {*boolean_function*}}

Specifies the partial state and its relevant Boolean function. This option is currently ignored in the PrimeTime tool.

-error_state {*state_name* {*boolean_function*}}

Specifies the error state and its relevant Boolean function. This option is currently ignored by the PrimeTime tool.

-ack_port {*port_name net_name* {*boolean_function*}}

Specifies the acknowledge port of the power switch and the logical net where this port connects. The command fails when,

- A port with the specified name already exists on the power switch.
- A net with the specified name does not exist. You can specify a Boolean function. This option is currently ignored by the PrimeTime tool.

-ack_delay {*port_name delay*}

Specifies the acknowledge port on the switch and the corresponding delay. This option is currently ignored by the PrimeTime tool.

-supply {*port_name supply_net_name*}

PT currently reads and ignores this option.

DESCRIPTION

This command enables you to create a power switch at the specified power domain. The switch is created within the scope of the power domain. Each power switch must be connected with an input supply net and an output supply net. The power switch can be connected with an acknowledge net and several control nets through a switch port. The switch ports are created if the power switch is created successfully, otherwise generates an appropriate error message.

EXAMPLES

The following example creates power switch, SW1 within power domain, PD1:

```
pt_shell> create_power_switch SW1 -domain PD1 \
          -output_supply_port {vout VNO2} \
          -input_supply_port {vin1 VNI1} \
          -control_port {ctrl_small ON1}  \
          -on_state {full_s vin1 {ctr_small}}
```

The following example generates an error message, when the command fails.

```
pt_shell> create_power_switch sw1 -domain ADD1 \
          -control_port {ctrl power_down} \
          -input_supply_port {in VDD1} \
          -output_supply_port {out VDD1_ADD} \
          -on_state {state_2001 in {!ctrl}}
Warning: Nothing implicitly matched 'ADD1' (SEL-003)
Error: Nothing matched for collection (SEL-005)
Error: The '-domain' option of create_power_switch command must specify
exactly one power domain object. (UPF-004)
0
```

SEE ALSO

get_power_switches(2)
report_power_switch(2)

create_qtm_constraint_arc

Creates a constraint arc for a quick timing model.

SYNTAX

```
string create_qtm_constraint_arc
   [-name arc_name]
   [-setup]
   [-hold]
   [-recovery]
   [-removal]
   -from port_name
   [-to port_spec]
   -edge rise | fall
   [-signal_edge rise | fall]
   [-path_type name]
   [-path_factor multiplication_factor]
   [-value constraint_value]
```

Data Types

arc_name	string
port_name	string
port_spec	list
name	string
multiplication_factor	float
constraint_value	float

ARGUMENTS

-name *arc_name*

Specifies the name of the constraint arc.

-setup

Creates a setup arc.

-hold

Creates a hold arc.

-recovery

Creates a recovery arc.

-removal

Creates a removal arc.

-from ***port_name***

Specifies the constraining port name. Define this port as a clock port.

-to ***port_spec***

Specifies the constrained port. Define this port as an input/inout port.

-edge rise | fall

Specifies the triggering edge of the clock (rise or fall).

-signal_edge rise | fall

Specifies the direction of the signal at the constrained port specified by the **-to** option.

If you do not specify this option, the tool creates constrinats for both signal rise and signal fall.

-path_type ***name***

Specifies the path type that you want to use to set the delay.

-path_factor ***multiplication_factor***

Specifies the multiplication factor for the path type.

-value ***constraint_value***

Specifies the delay value in terms of the time units you use for the model.

DESCRIPTION

This command allows you to create a constraint arc in a quick timing model. The **-from** port denotes the constraining port, and the **-to** port denotes the constrained port. The **-from** port must be defined as a clock port and the **-to** port must be an input or inout port. You can use **-signal_edge** option to specify the the direction of the signal at the constrained port specified by the **-to** option.

Use the **-setup**, **-hold**, **-recovery** and **-removal** options to specify a setup, hold, recovery or removal arc. You must use one of the switches, but you cannot use them simultaneously. To specify the triggering edge of the clock, use the **-edge** option and specify **rise** or **fall**.

To specify the delay value, use the **-path_type** option. You can specify the delay as a value in terms of the time units used in the design. Use either the **-path_type** or the **-value** option; you cannot use both simultaneously.

If the constraint arc is a setup/hold/recovery/removal arc, the quick timing model (QTM) parameter global setup/hold/recovery/removal time is added to the value of the specified type correspondingly. You must define the global setup/hold/recovery/removal time before creating any arc of the correspongin type. The command checks to see if these parameters are defined. Global setup/hold/recovery/removal time are defined using the **set_qtm_global_parameter** command.

To see the information about the current quick timing model, use the **report_qtm_model** command.

For a basic description about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command creates a setup arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
pt_shell> create_qtm_constraint_arc -setup -edge rise -from CLK -to IN1
            -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
pt_shell> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1
            -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay of 3 time units.

```
pt_shell> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1 -value 3.0
```

SEE ALSO

create_qtm_delay_arc(2)
create_qtm_model(2)
create_qtm_path_type(2)
remove_qtm_constraint_arc(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)

create_qtm_delay_arc

Creates a delay arc for a quick timing model (QTM).

SYNTAX

```
string create_qtm_delay_arc
   [-name arc_name]
   -from port_spec
   -to port_spec
   [-edge rise | fall]
   [-from_edge rise | fall]
   [-to_edge rise | fall]
   [-path_type path_type]
   [-path_factor multiplication_factor]
   [-value delay_value]
   [-total_value total_value]
```

Data Types

arc_name	string
port_spec	list
path_type	string
multiplication_factor	float
delay_value	float
total_value	float

ARGUMENTS

-name *arc_name*

Specifies the name of the delay arc.

-from *port_spec*

Specifies the QTM port name or a collection of QTM ports, which is the startpoint of the delay arc. For an edge triggering arc this must be a clock.

-to *port_spec*

Specifies the QTM port name or a collection of QTM ports. This port must be output or inout type.

-edge rise | fall

Specifies the triggering edge (rise or fall)

-from_edge rise | fall

Specifies the triggering edge (rise or fall) for clock start points or the signal direction for data startpoints. If you

do not use this option, the command creates delay arcs from both rise and fall edges at the start points.

-to_edge rise | fall

Specifies the signal direction at the data endpoints. If you do not use this option, the command creates delay arcs to both rise and fall edges at the endpoints.

If you omit both the **-from_edge** and **-to_edge** options, the command creates **rise-to-rise** and **fall-to-fall** delay arcs.

-path_type *path_type*

Specifies the path type you want to use to set the delay.

-path_factor *multiplication_factor*

Specifies the multiplication factor for the path type.

-value *delay_value*

Specifies the delay value in terms of the time units you use for the model.

-total_value *total_value*

Specifies the total delay value in terms of the time units used for the design. The drive arc delay and load are not considered when determining the total delay value for the arc.

DESCRIPTION

This command allows you to create a delay arc in a QTM from a QTM port or collection of QTM ports to a port or collection of QTM ports. If you specify a list of ports for the **-from** and **-to** ports, a delay arc is created from each start port to each end port.

To create an edge triggering arc, use the **-edge** option. For an edge triggering arc, the **clk_to_output** delay (specified as a global parameter) is added to the delay value specified. You can use **-from_edge** and **-to_edge** to specify the triggering edge for clock start points or the signal direction for data startpoints and endpoints.

You can use the **-path_type** option to specify the delay value or you can specify the delay as a value in terms of the time units used in the design. Use either the **-path_type** or the **-value** option; you cannot use both options simultaneously.

To show the information about the current QTM model, use the **report_qtm_model** command.

For a basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command creates a delay arc from input ports {A, B, C} to output port D with a delay value of 2.0 time units.

```
pt_shell> create_qtm_delay_arc -from {A, B, C} -to D -value 2.0
```

The following command creates an arc from clock CLK to output port OUT with a delay equivalent to 3 times that of path type 'path1'.

```
pt_shell> create_qtm_delay_arc -from CLK -to OUT -path_type path1 -path_factor 3
```

The following example uses wildcard to match all ports matching "CL*".

```
pt_shell> create_qtm_delay_arc -from CL* -to OUT -path_type path1 -path_factor 3
```

The following example uses a collection for the QTM port 'CLK':

```
pt_shell> create_qtm_delay_arc -from [get_qtm_ports CLK] -to OUT
          -path_type path1 -path_factor 3
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_model(2)
create_qtm_path_type(2)
remove_qtm_delay_arc(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)
```

create_qtm_drive_type

Creates a drive type in a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_drive_type
   -lib_cell lib_cell_name
   [-input_pin input_pin_name]
   [-output_pin output_pin_name]
   [-input_transition_rise rtrans]
   [-input_transition_fall ftrans]
   drive_type_name
```

Data Types

```
lib_cell_name    string
input_pin_name   string
output_pin_name  string
rtrans           float
ftrans           float
drive_type_name  string
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the library cell name in the technology library.

-input_pin *input_pin_name*

Specifies the input pin in the *lib_cell_name* is the startpoint of the arc and ends in the output pin.

-output_pin *output_pin_name*

Specifies the output pin in the *lib_cell_name* that specifies the drive.

-input_transition_rise *rtrans*

Specifies the input rising transition time associated with the **-input_pin** option to compute the delay for the drive arc. If you do not include this option, the delay table for rising input of the drive arc are loaded from library as is.

Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the *input_pin* option. This can obtain more accurate information on the transition time and delay time for the defined drive arc.

-input_transition_fall *ftrans*

Specifies the input falling transition time associated with the **-input_pin** option to compute the delay for the drive arc. If you do not include this option, the delay table for falling input of the drive arc are loaded from library as is.

drive_type_name

Specifies the name given to the defined drive type.

DESCRIPTION

This command can be used to create a drive type. The defined drive type can be used to set the drives on output ports. The drive type is specified by referring to a lib_cell in the technology library. The library cell is specified by using the **-lib_cell** option. This library cell must be present in the technology library.

The output pin, which drives the port, is specified by using the **-output_pin** option. In addition, you can specify the input pin by using the **-input_pin** option. This option selects an arc that drives the output pin of the library cell.

If neither the input pin nor the output pin is specified, an arbitrary arc is chosen to define the drive type. If the input pin is specified and output pin is not specified, an arbitrary arc starting from the input pin defines the drive type. If the output pin is specified and input pin is not specified, the arbitrary delay arc coming into the output pin defines the drive type. If both input and output pins are specified, the library cell must have an combinational arc from the input pin to the output pin.

To use this command, you must read in the technology library and then specify it by using the **set_qtm_technology** command.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For a basic description quick timing models, see the **create_qtm_model** man page.

EXAMPLES

In the following examples it is assumed that you have loaded the technology library, and the quick timing model technology is set. Do this by using the following the sequence of commands as a guide, substituting the name of your technology library.

```
pt_shell>read_db my_technology_library.db
```

```
pt_shell>set_qtm_technology -library my_technology_library
```

The following command creates a drive type equivalent to the BUF1 library cell.

```
pt_shell> create_qtm_drive_type -lib_cell BUF1 drive1
```

The following command creates a drive2 drive type equivalent to BUF2 library cell and specifies the output pin to use.

```
pt_shell> create_qtm_drive_type -lib_cell BUF2 -output_pin Y drive2
```

SEE ALSO

```
create_qtm_load_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
set_qtm_technology(2)
```

create_qtm_generated_clock

Creates a QTM generated clock.

SYNTAX

```
string create_qtm_generated_clock
   -source master_clock_name
   [-divide_by divide_factor | -multiply_by multiply_factor]
   [-invert]
   generated_clock_name
```

Data Types

master_clock_name	string
divide_factor	int
multiply_factor	int
generated_clock_name	string

ARGUMENTS

-source *master_clock_name*

Specifies the name of the clock defined as master source of the generated clock. The master source must be defined as a clock, or a generated clock, prior to the generated clock definition.

-divide_by *divide_factor*

Specifies the frequency division factor. If the *divide_factor* value is 2, the generated clock period is twice as long as the master clock period.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. If the *multiply_factor* value is *3*, the generated clock period is one-third as long as the master clock period.

-invert

Inverts the generated clock signal (in the case of frequency multiplication and division).

generated_clock_name

Specifies the name of the generated clock. If the name has been used in defining a port or internal pin, then the generated clock is defined on the existing port or internal pin. Otherwise, a new same-named internal pin is created to be the source of the generated clock.

DESCRIPTION

This command creates a QTM generated clock. A generated clock can be defined on the external port of the QTM, as well as the internal pin in the QTM. After defining the generated clock, the source port/pin can be used in other QTM commands the same way as those port/pins defined by the **create_qtm_port** command, i.e. delay arcs from or to the generated clock source pin can be defined with the **create_qtm_delay_arc** command; constraint arcs related to the generated clock can be defined with the **create_qtm_constraint_arc** command.

For a basic description about QTM, see the **create_qtm_model** man page.

EXAMPLES

The following example creates a divide-by 2 generated clock on an internal pin, assume there is no port defined with name "GCLK_int", with master clock CLK.

```
pt_shell> create_qtm_generated_clock GCLK_int -source CLK -multiply_by 2
```

The following example creates a generated clock on port GCLK, assume GCLK has already been defined as a port with command create_qtm_port.

```
pt_shell> create_qtm_generated_clock GCLK -source CLK -divide_by 2 -invert
```

SEE ALSO

```
create_qtm_model(2)
remove_qtm_generated_clock(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)
```

create_qtm_load_type

Creates a load type for a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_load_type
   -lib_cell lib_cell_name
   [-input_pin input_pin_name]
   load_type_name
```

Data Types

```
lib_cell_name       string
input_pin_name      string
load_type_name      string
```

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell in the technology library.

-input_pin *input_pin_name*

Specifies the input pin in the lib_cell to specify capacitance.

load_type_name

Specifies the name given to the defined load type.

DESCRIPTION

This command is used to create a load type. A load type can be used to define the load on a QTM input port. The load type is specified by referring to the *lib_cell* option in the technology library. The library cell is specified using the **-lib_cell** option. The library cell must be present in the specified technology library.

The input pin can be specified by using the **-input_pin** option. If the input pin is not specified, an arbitrary input pin is chosen to define the load type.

To use this command you must first read in the technology library, then specify the technology library by using the **set_qtm_technology** command.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

In the following example, the technology library is loaded, and the QTM technology is set:

```
pt_shell> read_db my_technology_library.db
pt_shell> set_qtm_technology -library my_technology_library
```

The following command creates a load type load1 using cell OR1. Since the input pin name is not specified, QTM selects an arbitrary input pin.

```
pt_shell> create_qtm_load_type -lib_cell OR1 load1
```

The following command creates a load type load2 using cell OR1, input pin A.

```
pt_shell> create_qtm_load_type -lib_cell OR1 -input_pin A load2
```

SEE ALSO

```
create_qtm_drive_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_load(2)
```

create_qtm_model

Begins the definition of a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_model
   model_name
```

Data Types

model_name	string

ARGUMENTS

model_name

Specifies the name of the generated model.

DESCRIPTION

Specifies the name of a new quick timing model (QTM) to be created by PrimeTime. QTMs are temporary timing models that can be created quickly for a block using PrimeTime commands. The purpose of these commands is to provide timing information without the necessity of writing a detailed timing model.

QTMs are intended to be used early in the design cycle to describe rough initial timing of a block. These models will eventually be replaced, either by some other detailed timing model or by the netlist of the block, to obtain more accurate timing.

A QTM can be saved as a Synopsys DB file, which can be instantiated in a design in the same way library cells or ITS models are instantiated. Both Design Compiler and PrimeTime accept designs with instantiated QTMs. To save a QTM model, use the **save_qtm_model** command.

Other commands in the QTM command set must be placed between a **create_qtm_model** and **save_qtm_model** command. Many QTM commands exist for specifying, configuring and examining QTMs. The following is not an exhaustive list, but gives examples of tasks you can perform using QTM commands:

- To create a QTM port, use the **create_qtm_port** command.
- To create constraint arcs and delay arcs, use the **create_qtm_constraint_arc** and the **create_qtm_delay_arc** commands, respectively.
- To set the drive of a QTM output port, use the **set_qtm_port_drive** command.

- To set the load of a QTM input port, use **set_qtm_port_load** command.
- To show the information about the current QTM model, use the **report_qtm_model** command.

EXAMPLES

The following command creates a new QTM model with the name adder.

```
pt_shell> create_qtm_model adder
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_path_type(2)
create_qtm_port(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_global_parameter(2)
set_qtm_port_drive(2)
set_qtm_port_load(2)
set_qtm_technology(2)
```

create_qtm_path_type

Creates a path type in a quick timing model (QTM) description.

SYNTAX

```
string create_qtm_path_type
   -lib_cell name
   [-input_pin pin_name]
   [-output_pin pin_name]
   [-fanout count]
   path_type_name
```

Data Types

name	string
pin_name	string
count	integer
path_type_name	string

ARGUMENTS

-lib_cell *name*

Specifies the name of the library cell in the technology library.

-input_pin *pin_name*

Specifies the input pin in the **-lib_cell** option, which is the startpoint of the arc, to define the path type.

-output_pin *pin_name*

Specifies the output pin in the **-lib_cell** option, which is the endpoint of the arc, to define the path type.

-fanout *count*

Specifies the average fanout (number of pins) to consider while computing the delay of the path type. The fanout count can range from 1 to 1000. The default is 1.

path_type_name

Specifies the name given to the defined path type.

DESCRIPTION

This command is used to create a path type. A path type mimics an arc in a library cell. An arc in the QTM model is defined in terms of the delays of the path type. The path type is specified by referring to a library cell in the technology library. The library cell is specified using the **-lib_cell** option. The library cell must be present in the technology library specified.

You can specify the average fanout count the arc fans out to while defining the path type. It is assumed the arc fans out to the first input pin of the same library cell, while calculating the delays. The input pin is specified by using the **-input_pin** option. The output pin is specified by using the **-output_pin** option.

If neither input pin nor the output pin is specified, an arbitrary delay arc from the library cell is chosen to define the path type.

If the input pin is specified and output pin is not specified, an arbitrary delay arc staring from the input pin defines the path type.

If the output pin is specified and input pin is not specified, an arbitrary delay arc coming into the output pin defines the path type.

If both input and output pins are specified, the library cell must have an arc from the input pin to the output pin.

To use this command you must read in the technology library then specify the technology library by using the **set_qtm_technology** command.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For a basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

In the following examples the technology library is loaded by the user and the QTM technology is set. This is done by the following sequence of commands:

```
pt_shell> read_db my_technology_library.db
```

```
pt_shell> set_qtm_technology -library my_technology_library
```

The following command creates a path type path1 by using lib_cell AN2, with an average fanout count of 2 (uses default input, output pins).

```
pt_shell> create_qtm_path_type -lib_cell AN2 -fanout 2 path1
```

The following command creates a path type path2 by using cell AN2, with an average fanout count of 2, and using pin 'A' as the starting point for the path type.

```
pt_shell> create_qtm_path_type -lib_cell AN2 -input_pin A -fanout 2 path2
```

The following command creates a path type path3 by using cell AN2, with an average fanout count of 2, using pin A as the start point of the arc, and pin Z as the endpoint of the arc.

```
pt_shell> create_qtm_path_type -lib_cell AN2 -input_pin A -output_pin Z -fanout 2 path3
```

SEE ALSO

create_qtm_constraint_arc(2)

```
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_model(2)
report_qtm_model(2)
save_qtm_model(2)
```

create_qtm_port

Creates a quick timing model (QTM) port.

SYNTAX

```
string create_qtm_port
   -type port_type
   port_list
```

Data Types

```
port_type     string
port_list     list
```

ARGUMENTS

-type *port_type*

Specifies the type of port. The port can be one of the following types: input, output, inout, internal or clock. If you want a port to be a clock, define it as a clock port.

port_list

Specifies the list of QTM ports you want created.

DESCRIPTION

This command creates QTM port. You can create a single port or a list of ports with this command. The ports can be input, output, inout, internal or clock. Any port that constrains another port or has an edge triggered launch arc originating from it, has to be defined to be of the type 'clock'.

Bused ports are also defined by giving the start and end index (A[0:5]). Bused ports cannot be internal.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following example creates an input bused QTM port A[0:5].

```
pt_shell> create_qtm_port A[0:5] -type input
```

The following example creates a clocked QTM port CLK.

```
pt_shell> create_qtm_port CLK -type clock
```

The following example creates QTM output ports A, B, C, and D.

```
pt_shell> create_qtm_port {A B C D} -type output
```

The following example creates QTM internal pin D_int.

```
pt_shell> create_qtm_port D_int -type internal
```

SEE ALSO

```
create_qtm_model(2)
remove_qtm_port(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
set_qtm_port_load(2)
```

create_rail_macro_model

Creates rail macro models to capture the analog effects in a hard macro. For more information about rail macro models, see the **rail_macro_model_flow** man page.

SYNTAX

```
status create_rail_macro_model
   -library library_name
   -cell cell_name
   [-format milkyway|ndm]
   [-tech_file file_name]
   [-gds_file file_name_list]
   [-layer_map_file file_name]
   [-power_nets power_net_name_list]
   [-ground_nets ground_net_name_list]
   [-extract_by gds_text|fram_view]
   [-tracing_file file_name]
   [-layer_text mask_name layer_number_list]
   [-allow_unmapped_layers]
   [-include_undefined_data_type]
   [-auto_exclude_bit_cells true|false]
   [-report_gds_arrays]
   [-skip_via via_mask_name_list]
   [-skip_conn_view_by_cell cell_name_list]
   [-stop_at_mask_layer mask_name]
   [-generate_current_source_location true|false]
   [-skip_current_source_by_cell cell_name_list]
   [-generate_switch_cell_location true|false]
   [-od_layer layer_number]
   [-switch_cell_output_supply_nets supply_net_name_list]
   [-switch_cell_on_resistance resistance_value]
   [-ignore_foreign true|false]
   [-reuse_conn_view]
   [-hspice_exec hspice_path]
   [-spice_top_netlist file_name]
   [-spice_model_config file_name]
   [-spice_include_path path_list]
   [-spice_model_condition process_corner process voltage temperature]
   [-spice_net_map_file file_name]
   [-si_ware_project_path project_path cell_name]
   [-layer_translation_table file_name]
   [-switch_supply_net_pairs net_pair_list]
   [-mode_configure file_name]
```

Data Types

```
string library_name
string cell_name
string file_name
string mask_name
string hspice_path
string process_corner
string project_path
```

```
float resistance_value
float process
float voltage
float temperature
list file_name_list string
list power_net_name_list string
list ground_net_name_list string
list mask_name_layer_number_list string integer
list via_mask_name_list string
list cell_name_list string
list supply_net_name_list string
list path_list string
list net_pair_list string string
integer layer_number
```

ARGUMENTS

-library *library_name*

Specifies the Milkyway design library where the Milkyway CEL view resides. When used with the -gds_file option, specify the name of the Milkyway library to be created. *library_name* must be valid if -format *ndm* is used.

-cell *cell_name*

Specifies the name of the Milkyway CEL design to be opened. When used with the -gds_file option, specify the name of the cell to be created in the Milkyway library. If the cell list is given, please specify -auto_exclude_bit_cells FALSE. The -auto_exclude_bit_cells option does not support to handle mutilple cells

-format *milkyway|ndm*

Specifies library format. Milkyway is default setting.

-tech_file *file_name*

Specifies the technology file for macro model creation. Use this option in combination with -gds_file option.

-gds_file *file_name_list*

Specifies GDSII files. This option is mandatory if -format *ndm* is used.

-layer_map_file *file_name*

Specifies the layer mapping file which describes the layer mapping between GDSII and technology files. Use this option in combination with the -gds_file option.

Following is an example of a layer mapping file:

```
; MilkywayLayer:MilkywayDataType  GDSIILayer:GDSIIDataType
6:0  6:0
10   6:7   ; map GDSIILayer 6, GDSIIDataType 7 to MilkywayLayer 10, MilkywayDataType 0
17:0 17:0
17:7 17:7
18:5 17:11 ; map GDSIILayer 17, GDSIIDataType 11 to MilkywayLayer 18, MilkywayDataType 5
31:0 31    ; map GDSIILayer 31, any GDSIIDataType to MilkywayLayer 31, MilkywayDataType 0
31:7 32:7
```

When a layer mapping file is unavailable, the command invokes the direct-mapping mechanism for CONN view extraction and issues the following warning message:

```
Warning: No layer mapping file defined. By default, the GDSII
         layer:dataType to Milkyway layer:dataType mapping is
         direct and uses the same layer numbers.
```

If the specified layer mapping file contains incomplete layer data, that is, not all the GDSII layer:dataType layers are defined in the layer mapping file, the command issues a warning message for those undefined GDSII layer:dataType layers:

```
Warning: Found unmapped layer (3:0). Please correct your layer
         mapping file.
Warning: Found unmapped layer (17:12). Please correct your layer
         mapping file.
```

-power_nets *power_net_name_list*

Specifies the names of the top-level power nets to be extracted. When the -extract_by *gds_text* option is specified, you must provide either power or ground net names for a successful run.

-ground_nets *ground_net_name_list*

Specifies the names of the top-level ground nets to be extracted. When the -extract_by *gds_text* option is specified, you must provide either power or ground net names for a successful run.

-extract_by *gds_text|fram_view*

Uses the text in the GDSII file or the pin shapes in the FRAM view as the tracing source for CONN view extraction. The default setting is *gds_text*.

-tracing_file *file_name*

Specifies the file in which the tracing sources for CONN view extraction are defined. When a tracing file is used, the tracing sources from the GDSII text or the pin shapes from the FRAM view will be ignored.

Define tracing sources in a tracing file in the following syntax: pg_net_name mask_name x_in_Milkyway_unit y_in_Milkyway_unit

Following is an example of the tracing file:

```
# <pg_net_name> <mask_name> <x_in_Milkyway_unit> <y_in_Milkyway_unit>
VDD metal4 100 10
VSS metal3 22 1
```

-layer_text *mask_name_layer_number_list*

Specifies the metal mask names with the name or number of the text layers that identify the corresponding metal layer geometries. If all text layers match the geometry layers, deselect this option.

The format of the layer text is {mask_name layer_number layer_number ...}

For example, to use layers 32 and 33 to identify the metal2 geometry, and layer 31 to identify the metal3 geometry, specify the option in the following syntax:

```
pr_shell> create_rail_macro_model -layer_text {{metal2 32 33} {metal3 31}}
```

-allow_unmapped_layers

During CONN view extraction, directly maps the GDSII layer:dataType entries that are not defined in the layer mapping file to those in the technology file. Use this option when the specified layer mapping file contains incomplete layer information and direct mapping has no conflict with the technology file.

-include_undefined_data_type

Includes the shapes of undefined data types during CONN view extraction. This option is valid only when the -

allow_unmapped_layers option is specified.

For example, if the specified GDS, layer mapping and technology files have the following information:

```
The GDSII file:
  31:0
  31:5
  31:10
  31:25

The layer mapping file:
  ; MilkywayLayer:MilkywayDataType  GDSIILayer:GDSIIDataType
  31:0      31:0
  31:5      31:5

The technology file:
  31:0
  31:5
  31:25

Then, when running the command without the -allow_unmapped_layers or
-include_undefined_data_type option, the extracted CONN view contains
the following information:
  31:0
  31:5

If both the -allow_unmapped_layers and -include_undefined_data_type options
are specified, the extracted CONN view has the following information:
  31:0
  31:5
  31:10

The entry 31:25 is excluded because it is already defined in the
technology file.
```

-auto_exclude_bit_cells *true*|*false*

Automatically identifies and excludes bit cells when generating rail macro models for a memory cell. The default setting is true. PrimeRail identifies a bit cell when the cell has more than 100,000 instances and those instances occupy more than 50% of the chip area. If no cell meets the criteria, PrimeRail reports no bit cell is identified with the following message:

```
INFO: No bit cell was identified and automatic bit cell skipping will not be launched.
```

When multiple cells meet the criteria, PrimeRail skips the one with the highest instantiation. The following message is printed when a bit cell is found and skipped during rail macro model generation:

```
INFO: Cell BC_TEST is identified as a bit-cell hierarchy and is being skipped during
Rail-Macro-Model creation.
```

-report_gds_arrays

Lists possible bit cells that meet the default criteria before skipping them during macro model generation. To skip a bit cell reported in the list, rerun the **create_rail_macro_model** command with -skip_conn_view_by_cell *cel_name* option.

Following is an example of the output report:

```
================================================================================
Candidate Cell        #Inst on IP    %Area of IP   Layer in Cell
================================================================================
difcon                 200000         84.95%        Metal1
VIA                    160000         67.96%        Metal2 Via1 Metal1
polycon                120000         50.97%        Metal1
```

-skip_via *via_mask_name_list*

Avoids extracting the connectivity through the specified vias. By default, the command extracts the connectivity from via1 to via14 (that is, from metal1 to metal15).

-skip_conn_view_by_cell *cell_name_list*

Excludes the specified child cells from the CONN view extraction. By default, all child cells are included.

-stop_at_mask_layer *mask_name*

Stops generating rail macro models at the specified mask layer. The generated rail macro model contains the data from the top mask layer to the specified mask layer.

-generate_current_source_location *true|false*

Chooses to generate current source files for the specified design during macro model generation. By default, the command generates current source files and then attaches them to the CONN view. With the current source files, PrimeRail places current sinks for hard macros at the locations based on the current source files. This is to improve the rail analysis accuracy.

-skip_current_source_by_cell *cell_name_list*

Excludes the specified child cells from the current source file generation. By default, all child cells are included.

-generate_switch_cell_location *true|false*

Chooses to generate a switch cell location file for the specified design during CONN view extraction and to attach the file to the CONN view. The default setting is false.

The switch cell location file records on-resistance locations inside the cell and is used during the power-on analysis flow.

-od_layer *layer_number*

Specifies the OD layer number to identify a device during switch cell extraction.

-switch_cell_output_supply_nets *supply_net_name_list*

Specifies the output supply net names for switch cells. You must specify virtual supply net names when extracting switch cells.

-switch_cell_on_resistance *resistance_value*

Specifies the on-resistance value for modeling switch cells during static rail analysis.

-ignore_foreign *true|false*

Specifies if not taking FOREIGN property from LEF import into consideration. The default setting is false; that means FOREIGN property will be used.

-reuse_conn_view

Reuses the CONN view when updating an existing rail macro model. When specified, only the electrical data will be updated for the rail macro model being updated.

-hspice_exec *hspice_path*

Specifies the path to the HSPICE binary for simulating the circuit when creating rail macro models with SPICE data.

-spice_top_netlist *file_name*

Specifies the top-level SPICE netlist if the top circuit cannot be found automatically. This top-level SPICE netlist contains the device location information that is generated by an LVS tool. The file must contain the $X, $Y and $T keywords.

Following is an example of the netlist that is generated by the IC Validator tool.

> * An ICV Netlist Example \[char46]subckt top X1 n0 n1 n2 test $T=0 0 0 0 $X=0 $Y=0 X2 n3 n4 n5 test $T=70 0 0 90 $X=300 $Y=90 \[char46]ends \[char46]subckt test M0 a b c 1 nch $X=20 $Y=30 $D=0 \[char46]ends

-spice_model_config *file_name*

Specifies the name of the SPICE model configuration file which contains the setup information for the SPICE simulation.

Here is an example of the configuration file:

```
\[char46]LIB SSQU
\[char46]lib 'device_model.sp' SS_SRAM
\[char46]lib 'device_model.sp' SSMacro_MOS_MOSCAP
\[char46]ENDL
\[char46]LIB TTQU
\[char46]..
\[char46]ENDL
```

-spice_include_path *path_list*

Specifies paths including SPICE netlists.

-spice_model_condition *process_corner process voltage temperature*

Specifies the SPICE simulation process corner. By default, the voltage corner is used for SPICE characterization.

-spice_net_map_file *file_name*

Specifies the net mapping between the GDSII file and the SPICE netlist.

-si_ware_project_path *project_path cell_name*

Specifies the full hierarchical path to where the memory compiler is installed and the cell name to be modeled.

-layer_translation_table *file_name*

Specifies the layer translation table file for generating a layer mapping file. This translation table file is saved in the memory compiler library and contains only the minimum set of the mapping information.

When both *layer_map_file* and *layer_translation_table* are specified, the *layer_map_file* takes the precedence.

-switch_supply_net_pairs *net_pair_list*

Specifies the internal and external net pairs of fine-grain memory cells, in the following syntax:

```
pr_shell> create_rail_macro_model -switch_supply_net_pairs {{internal_net external_net}}
```

-mode_configure *file_name*

Specifies the file which defines mode-related data for macro models. Each mode definition is composed of the following information: name: mode name waveform_source: FSDB file containing current waveforms pg_pin: PG pin name related_pin: trigger pin when: when condition start_time: time to start to capture current waveforms end_time: time to stop capturing current waveforms strip: path prefix to be stripped

The mode configuration file should also define the signal pins listed in *related_pin*. Each definition is composed of the following information: signal: signal pin name dir: signal direction; can be either input, output, or inout

Following is an example of the mode configuration file:

```
# Comment '#' and concatenated line via '\' are supported.
# No space is allowed in the argument assignment.
# For example, "name:VDD" is supported, but "name: VDD" is not.

# Defining signal pins
signal:MEA  dir:input
signal:WEA  dir:input
signal:RMEA dir:input
signal:LS2  dir:input
signal:CLKA dir:input
signal:MEB  dir:input
signal:RMEB dir:input
signal:LS   dir:input
signal:CLKB dir:input

# Supporting multiple modes with different time periods inside one FSDB file
name:W0 start_time:1125 end_time:1625 \
  waveform_source:xa.fsdb \
  pg_pin:VDD strip:"x_testbench." when:"MEA&WEA&!RMEA&!LS2" related_pin:"CLKA"
name:R0 start_time:2375 end_time:2875 \
  waveform_source:xa.fsdb \
  pg_pin:VDD strip:"x_testbench." when:"MEB&!RMEB&!LS" related_pin:"CLKB"

# Supporting multiple FSDB files for different modes
name:W0 start_time:1125 end_time:1625 \
  waveform_source:xa2.fsdb \
  pg_pin:VSS strip:"x_testbench." when:"MEA&WEA&!RMEA&!LS2" related_pin:"CLKA"
name:R0 start_time:2375 end_time:2875 \
  waveform_source:xa3.fsdb \
  pg_pin:VSS strip:"x_testbench." when:"MEB&!RMEB&!LS" related_pin:"CLKB"

# Any duplicate name or pg_pin combination is illegal. Only the first one will
# be honored.
name:R0 start_time:5375 end_time:5875 \
  waveform_source:xa.fsdb \
  pg_pin:VSS strip:"x_testbench." when:"MEB&!RMEB&!LS" related_pin:"CLKB"
```

DESCRIPTION

This command creates rail macro models based on the specified data. When both the FRAM and CEL views are available for the target design, the command creates rail macro models by extracting geometry information from the library, and rebuilds net information for power and ground nets. The stream-in CEL view must contain the net or pin text information.

When the stream-in CEL view is unavailable, you can create the rail macro models with the GDSII data, along with the layer mapping and technology files.

To improve the accuracy of the rail macro model, you can create the rail macro models with the SPICE files, including a SPICE netlist and a configuration file. You must specify the path to the simulation tool, such as HSPICE, for simulating the circuit.

If the macro model is saved to the Milkyway library, make sure the library is writable.

To examine the content of the generated rail macro model, use the **report_rail_macro_model** command.

EXAMPLES

The following example generates a rail macro model in which current is evenly distributed by reading PG pin shapes from the sram64x64.gds file.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -layer_map_file layer.map -tech_file tech.tf \
            -gds_file sram64x64.gds -power_nets VDD -ground_nets VSS

pr_shell> report_rail_macro_model -library test_lib -cell sram64x64
```

The following example generates a rail macro model with the GDSII file named sram64x64.gds. The FRAM view for the cell sram64x64 is also provided. The example reads and extracts all power and ground nets in the Milkyway library test_lib by using the pin shapes as the tracing source. The pin shapes are from the FRAM view.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -gds_file sram64x64.gds -extract_by fram_view
```

The following example uses the memory compiler outputs to generate a rail macro model for the cell sram64x64 in the Milkyway library si_ware_pr_rmm. All supply nets of sram64x64 are modeled.

```
pr_shell> create_rail_macro_model -si_ware_project_path \
            {/home/si_ware/project sram64x64} -tech_file mw.tf
```

The following example generates a rail macro model which contains the weighted current and capacitance distribution. The PG pin shapes are read from the CEL view named sram64x64 in the Milkyway library test_lib.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -power_nets VDD -ground_nets VSS \
            -spice_top_netlist sram64x64_LVS.sp \
            -spice_model_config config.sp \
            -spice_model_condition {SSQU 1.0 0.7 125.0} \
            -hspice_exec {/tools/hspice}

pr_shell> report_rail_macro_model -library test_lib -cell sram64x64
```

The following example generates a rail macro model containing internal-pin current waveforms and weighted capacitance distributions, based on the setting specified by the -mode_configure option. The PG pin shapes are from the CEL view in the name sram64x64 in the Milkyway library named test_lib.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -reuse_conn_view \
            -power_nets {VDD} -ground_nets {VSS} \
            -mode_configure mode_cfg \
            -spice_top_netlist sram64x64_LVS.sp \
            -spice_model_config config.sp \
            -spice_model_condition {SSQU 1.0 0.7 125.0} \
            -hspice_exec {/tools/hspice}
```

The following example generates the rail macro model with on-state resistances. The name of the internal net is vddpi and its related external net name is VDD. The hard macro is a fine-grain memory. The internal net cannot be viewed in the FRAM view.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -layer_map_file layer.map -tech_file tech.tf \
            -gds_file sram64x64.gds -power_nets VDD -ground_nets VSS \
            -spice_top_netlist sram64x64_LVS.sp \
            -spice_model_config config.sp \
            -spice_model_condition {SSQU 1.0 0.7 125.0} \
            -hspice_exec {/tools/hspice} \
            -switch_supply_net_pairs {{vddpi VDD}}
```

The following example generates a rail macro model for the cell sram64x64 in the Milkyway library test_lib. During the model generation, the command skips child cells SoftBlockA and SoftBlockB and stops the generation process at the metal2 layer.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -power_nets VDD -stop_at_mask_layer metal2 \
            -skip_conn_view_by_cell {SoftBlockA SoftBlockB}
```

The following example generates the rail macro model and switch cell location files for the cell switchA in the Milkyway library test_lib. The command extracts the power net VDD and the virtual power net VDDV. The OD layer is 17 and the on-resistance value is 3.4 ohm. The hard macro is a switch cell. The virtual net VDDV can be viewed in the FRAM view.

```
pr_shell> create_rail_macro_model -library test_lib -cell switchA \
            -power_nets VDD -generate_switch_cell_location true \
            -switch_cell_output_supply_nets VDDV \
            -od_layer 17 -switch_cell_on_resistance 3.4
```

The following example generates a rail macro model with automatic bit-cell exclusion for a memory hard macro.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -layer_map_file layer.map -tech_file tech.tf \
            -gds_file sram64x64.gds -power_nets VDD -ground_nets VSS

pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -layer_map_file layer.map -tech_file tech.tf \
            -gds_file sram64x64.gds -power_nets VDD -ground_nets VSS \
            -report_gds_arrays
```

The following example generates a rail macro model using tracing points defined in the tracing file trace.txt, instead of the GDSII texts or FRAM view pins.

```
pr_shell> create_rail_macro_model -library test_lib -cell sram64x64 \
            -layer_map_file layer.map -tech_file tech.tf \
            -gds_file sram64x64.gds -power_nets VDD -ground_nets VSS \
            -tracing_file trace.txt
```

SEE ALSO

rail_macro_model_flow(2)
report_rail_macro_model(2)
set_rail_macro_model_cells(2)

create_si_context

Generates an SI context for selected blocks of the design. A top level design and full chip binary parasitics can also be generated.

SYNTAX

```
status create_si_context
   [-include include_list]
   [-instances instance_list]
   [-parasitics_options para_options]
   [-top_inst instance_name]
   [-no_design_parasitics]
```

Data Types

```
include_list      list
instance_list     list
para_options      list
instance_name     string
```

ARGUMENTS

-include *include_list*

If this option is not specified, the context script will be generated for cross-talk. You can use this option to specify PrimeTime SI to generate the context scripts for cross-talk, noise or both. Allowed values are *xtalk* and *noise*. The generated context will be the same, only the script will be different.

-instances *instance_list*

Indicates that PrimeTime SI is to extract contexts for the given list of instances or for all the top level instances of the design.

-parasitics_options *para_options*

Specifies that a parasitic file has to be generated to be used to annotate the (block + context) design. The allowed values for the *para_options* options are *spef_format* and *sbpf_format*.

-top_inst *instance_name*

Specifies that PrimeTime should generate a top level design for the given instance. By default, PrimeTime generates top level design for the given design. This option can be used in conjunction with the *-instances* option to generate contexts and top level design for an arbitrary hierarchy. Note that all the instances provided in the *-instances* option must be the child instances of the top level instance provided in this option.

-no_design_parasitics

By default, this command also generates full chip parasitics in SBPF format. Use this option to indicate PrimeTime not to generate the SBPF file. This option should be used if your parasitic extraction tool generated SBPF.

DESCRIPTION

The **create_si_context** command generates contexts for all top level instances of the design or for the list of instances given in the *-instances* option. This context can be used to perform signal integrity analysis at the block level.

The effect of context on the block is the same as that of the top level and other blocks on the block. Hence, context enables accurate signal integrity analysis for block level analysis as if the block is being analyzed in the context of the full chip.

The command also generates a top level design and a top level script that can be used to perform chip level analysis after block level analysis is complete and respective ILMs are generated for each block. User can direct PrimeTime to generate contexts and top level design for an arbitrary level of hierarchy instance using the *-top_inst* and *-instances* options.

The command creates one directory for each block (for which a context is being extracted) and outputs the context files inside the directory. Each directory contains a verilog context design (named wrapper.v), a TCL script (named wrapper.tcl) that can be used in performing block level analysis, a text file that contains aggressor annotation points (named wrapper.txt) that can be used later for annotating arrivals/slews for accurate block level analysis. The command also generates an infinite arrival script (named wrapper.pt.gz) that can be used to perform conservative block analysis in the context. The command can generate this arrival script for cross-talk, noise or both depending on the *-include* option. In addition, if the *-parasitics_options* option is given, a parasitic file (named wrapper.sbpf or wrapper.spef.gz) gets generated. Also at the level of this new directory, a top level verilog file (named design.v), a top level TCL script (named design.tcl) are also generated. In addition, if the *-no_design_parasitics* option is not given, a full chip binary parasitic file (named design.sbpf) gets generated.

The **create_si_context** command is affected by the **pt_ilm_dir** environment variable. The top design files and directories for individual blocks are created at the location given by the **pt_ilm_dir** variable.

EXAMPLES

The following example generates contexts for all top level designs along with context parasitics in SBPF and a top level design.

```
pt_shell> create_si_context -parasitics_options {sbpf_format}
```

The following example generates contexts for instances "blockA" and "blockB" along with context parasitics in SPEF.

```
pt_shell> create_si_context -instances {blockA blockB} \
-parasitics_options {spef_format}
```

SEE ALSO

```
pt_ilm_dir(3)
write_arrival_annotations(2)
create_ilm(2)
```

create_supply_net

Creates a supply net defined for the specified power domain. The supply net is created in the logic hierarchy at the same scope as the specified power domain.

SYNTAX

```
string create_supply_net
   [-domain domain_name]
   [-reuse]
   [-resolve unresolved | parallel | one_hot | parallel_one_hot]
   supply_net_name
```

Data Types

```
domain_name         string
supply_net_name     string
```

ARGUMENTS

-domain *domain_name*

Specify which power domain this supply_net is defined for. If specified, the power domain must exist. If used with the **-reuse** option, the *supply_net_name* must correspond to a supply net which is already domain dependent.

-reuse

Reuse an existing supply net instead of creating a new one. You can associate one supply net with several power domains. If this option is specified, the supply net must exist.

-resolve unresolved | parallel | one_hot | parallel_one_hot

Resolution mechanism that determines the state and voltage of the supply net. This is intended also for checking allowed connectivity of supply nets through switches. PrimeTime currently accepts and stores this option but does not use it.

supply_net_name

Specify the name of the supply net to be created. The name should be a simple (nonhierarchical) name.

In the scope of the specified power_domain, or in the scope of the current instance if no power_domain is specified, if there is a supply net, logical hierarchical net, or logical port with the same name, the *supply_net_name* cannot be created. One exception is that when you use the **-reuse** option, the name must be the same with another existing supply_net in the same scope, and no checking is applied in this situation.

This argument is required.

DESCRIPTION

The **create_supply_net** command enables you to create a supply net defined in the specified power domain. A supply net connects supply ports and pins.

Each power domain has a primary power supply net and a primary ground supply net, and it could have several other supply nets. If the command succeeds, a supply net is created in the scope of power domain, or in the current scope, if no power domain is specified. It is currently neither primary power net nor primary ground net of the power domain. Use the **set_domain_supply_net** command to set it as the primary power/ground net.

Supply nets can be created either domain dependent, or domain independent (with or without the **-domain** option). If a supply net has been created domain independent, it cannot be changed to a domain dependent supply net by the use of the **-reuse** option in a later invocation of **create_supply_net**.

This command returns the full name of the supply net (from the current scope), if successful. It returns null string if it fails.

EXAMPLES

The following example creates a supply_net A_VDD in power_domain PD1, and also re-creates A_DD in another power_domain PD2 using the **-reuse** option:

```
prompt> create_power_domain PD1 -element INST1
PD1
prompt> create_power_domain PD2 -element INST2
PD2
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VDD -domain PD2 -reuse
A_VDD
prompt> create_supply_net B
B
```

SEE ALSO

```
create_power_domain(2)
report_supply_net(2)
set_domain_supply_net(2)
```

create_supply_port

Creates a supply port in specified power domain or in current scope if no power domain is specified.

SYNTAX

```
string create_supply_port
   supply_port_name
   [-domain domain_name]
   [-direction <in|out>]
```

Data Types

```
supply_port_name     string
domain_name     string
```

ARGUMENTS

supply_port_name

Specify the name of the supply port to be created. The name should be a simple (non-hierarchical) name. This argument can be used with -domain option to help avoiding hierarchical names.

In the scope of specified power_domain, if there is a supply_port, logical port, or logical hierarchical net with the same name, the supply_port cannot be created.

This argument is required.

-domain *domain_name*

Specify which power_domain where this port defines a supply net connection point. The power domain must exists in the current scope.

-direction <in|out>

Defines how state information is propagated through the supply network as it is connected to the port. If the port is an input port, the state information of the external supply net connected to the port shall be propagated into the domain. Likewise, for an output port, the state information of the internal supply net connected to the port shall be propagated outside of the domain.

The default value is in.

DESCRIPTION

The *create_supply_port* command enables you to create a supply port at the scope of the specified power domain or at the current scope. A supply port provides connection point for the supply net.

This command returns the supply_port object upon success. It returns null string if it failed.

EXAMPLES

The following example creates a supply port VN1 at the scope of the PD1 power domain:

```
prompt> create_power_domain PD1 -element INST1
PD1
prompt> create_supply_port VN1 -domain PD1
VN1
```

SEE ALSO

```
create_supply_net(2)
create_power_domain(2)
get_supply_ports(2)
```

create_taps

Creates tap point voltage source objects to be used in the subsequent rail analysis.

SYNTAX

```
status create_taps
   [-point {X-coordY-coord}]
   [-layer number_or_name]
   [-supply_net supply_nets]
   [-name tap_name]
   [-top_pg]
   [-of_objects object_list]
   [-import file_name]
   [-nocheck]
   [-result result_type]
   [-snap_distance snap_distance]
   [-honor_non_pg_pin]
```

Data Types

```
real X-coord
real Y-coord
string number_or_name
string tap_name
list supply_nets
string spiceFile
string file_name
list object_list
string result_type
real snap_distance
```

ARGUMENTS

-point {*X-coord Y-coord*}

Defines the location where the tap is to be created, in terms of the *X-coord* and *Y-coord* location on the XY plane. Use this option with the -layer option to define a physical location in the three-dimensional space at which the tap is to be created. When the -point option is used with the -of_objects option, the XY location is relative to the origin of the specified objects. Otherwise, it is relative to the origin of the current top-level Milkyway cell. This option is mutually exclusive with the -top_pg option.

If a tap point is defined in a location where a tap already exists, the tool issues a warning message and replaces the original one with the new one.

-layer *number_or_name*

Defines the layer at which the tap point is to be created. The layer can be specified as either a Milkyway layer

number or a layer name. If the -top_pg option is not specified, this option must be accompanied by the -point option. When used together, the -layer and -point options define a physical location in the three-dimensional space at which the tap is to be created.

-supply_net *supply_nets*

Specifies the supply net with which this tap point is associated. If both the -point and -layer options are used, this option is required and only one supply net can be specified. If the *object_list parameter* with the -top_pg option is specified along with the -layer and -point options, specifying this option is not required and a collection of supply net names is acceptable.

-name *tap_name*

Specifies the name for the tap point to be created. If the specified *tap_name* already exists in the design, the tool issues a warning message and replaces the original tap with the new one. When multiple tap points are created by one invocation of the **create_taps** command (only supported in Milkyway flow), all the created taps are named in the form *tap_name_NNN*, where NNN is a unique integer identifier and *tap_name* is the string argument to the -name option.

-top_pg

Indicates that tap points are to be created for all the top-level supply (power or ground) pins of Milkyway design cells. This argument cannot be used with the -point option. If the -layer option is specified, tap points are to be created at the specified layer.

-of_objects *objects_list*

Creates tap points on the specified objects. Each object can be a cell, a lib_cell, or a collection of cells and lib_cells. By default, the taps are created on the supply power or ground pins of the *objects*. If the -point option is specified, the command creates taps at the specified physical location that is relative to the origin of each object. In NDM flow, this feature is limited to top-level objects only.

-import *file_name*

Reads in the information from a file that defines tap X/Y locations, layer numbers and supply nets. The file can be either in the PrimeRail tap file format or the IC Compiler virtual power pad format. The command automatically detects the file format.

The supported PrimeRail tap file format is as follows:

<net_name> <layer_number> <X-coord> <Y-coord> [*R(<R_in_Ohm>)*] [*L(<L_in_Henry>)*] [*C(<C_in_Farad>)*]

If the tap file is used with a SPICE package file (loaded by the **set_tap_package_model -spice** command), the tap file format is as follows:

<net_name> <layer_number> <X-coord> <Y-coord> PKG_PORT *<package_model_port>*

The format of the supported IC Compiler virtual power pad file is as follows:

<net_name> <X-coord> <Y-*coord*> *<layer_number> <net_name> <X-coord>* <Y-*coord*> *<layer_number>*

-nocheck

Indicates that tap points are not checked to see if the tap points touch any layout geometry. This option has to be used if a tap point is defined inside a hard macro. By default, the command checks tap points against layout geometry if they are created based on the -point, -layer and -import options. For tap points created with this option, validity can be checked later with the **check_taps**, **calculate_minimum_path_resistance**, or **calculate_rail_voltage** command.

-result *result_type*

Imports the taps from the result context into the session context. The *result_type* can be either the *rail* or *integrity* keywords, which indicates whether the result context is rail or integrity result context.

The -result option is mutually excluded with the -of_object, -import, -top_pg, -point or -layer option.

See the **tap_contexts** man page for a detailed description about tap contexts.

-snap_distance *snap_distance*

Defines the snap distance in microns. When a given tap location does not touch a layout geometry of a supply net, PrimeRail searches for legal locations within the specified distance by expanding to the left, right, bottom and top corners. PrimeRail creates a new tap point at the corner that is within the minimum distance to the original location specified by the -point option.

The -snap option can only be used with either the -point or -import option.

-honor_non_pg_pin

Allows taps to be created on non-PG pins that are connected to a specified supply net when taps are created using the -of_object option.

DESCRIPTION

The **create_taps** command creates tap point objects in the current session, which is to be considered as part of supply network in the subsequent rail analyses.

Taps are used to model the external voltage source environment in which the device under analysis (DUA) operates. They are not part of the design itself, but can be thought of as virtual models of voltage sources. For some analyses, taps are treated only as sources of ideal voltage. For other analyses, taps represent the package in which the DUA is embedded. In these cases the electrical behavior can be significantly more complex than a simple ideal voltage source.

The **set_tap_package_model** command provides a mechanism for applying relatively simple electrical characteristics to the taps but with more complex behavior descriptions, such as when to deal with coupling between tap points, the tap point locations and electrical characteristics in a SPICE file.

A tap point is always associated with a specific net either explicitly or implicitly. If the **create_taps** command is run without the -supply_net option (which is allowed for the -top_pg and the -of_object options), the tap is implicitly associated to the supply (power or ground) net that is connected to the object from which the tap is defined.

When a physical location for a tap object is specified via the -point and -layer options, the tap is virtually connected to the design at that location. If there is no supply (or supply pin) net or via shape at the specified location, which means there is no conductive path to the supply network, the tap has no rail effect.

By default, the physical location specified by the -point and -layer options must touch a geometry as checking is done to ensure this. The location checking status is updated by setting the is_valid attribute from an initial unknown setting. You can disable the checking with the -nocheck option. For tap points defined inside a hard macro, user has to use -nocheck option to skip validity check against layout. When checking is enabled, if a defined tap point does not touch any layout geometry, a warning message is issued.

The command only checks top 100 tap points if they are specified through the -point and -layer options. When the design has more than 100 tap points to be checked, group them into a tap file and import it with the -import option. Alternatively, you can set the *rail_force_tap_check* variable to true to enforce the checking validation beyond the limit of 100.

When the -nocheck option is specified, the setting of the *rail_force_tap_check* variable is not honored.

If a tap is already present at a location at the time when the **create_taps** command is executed, the newly created tap replaces the existing tap. The previously defined tap point name is no longer valid. When this occurs, a summary message is issued, reporting the number of replaced tap points.

Two or more tap points might be created close to one another, typically through the application of multiple execution of the **create_taps** command to specify taps in different ways. If taps are placed too close to one another, the effective resistance between them might be too small to have an impact on the rail result. This means these taps effectively act as one single tap point. For example, if the taps are sufficiently separated at opposite ends of a relatively long segment of a net shape, there is non-trivial resistance between them, which alters the rail results more substantially.

When a *lib_cell* is provided within the *object_list* parameter, the tap points are created on all instances of that *lib_cell* used in the design.

All taps are uniquely named, either by using the -name option or by the default naming. If the -name option is not specified, the taps are named TAP_NNN, where NNN is a number that starts at 1 and is incremented with each created tap.

In some circumstances the built-in models available with the **set_tap_package_model -R -L -C** command are not sufficient to adequately describe the electrical behavior of the device package. This is especially true when coupling between the package inputs needs to be considered. In this case you need to use the **set_tap_package_model -spice** command to define the SPICE package model.

See the related chapter in the PrimeRail User Guide for details on how to write and use a package SPICE model.

If you have an ASCII file where describes information of taps to be created, use the -import option to read in this file and create taps accordingly. The file can be in the PrimeRail tap file format or IC Compiler virtual power pad file format.

EXAMPLES

This example creates taps for each instance of the lib_cell *some_pad_cell*. One tap per instance is created on metal1 and at a location of (30.4 16.3) relative to the origin of the cell instance.

```
pr_shell> create_taps -of_objects [get_lib_cell */some_pad_cell] -layer metal1 -point {30.4 16.3}
```

This example creates taps on all power or ground pins of all cell instances matching the pattern **power_pad_cell**.

```
pr_shell> create_taps -of_objects [get_cells -hier *power_pad_cell*]
```

The following example creates a tap using absolute coordinates. The command checks whether it touches any layout geometry.

```
pr_shell> create_taps -supply VDD -layer 3 -point {100.0 200.0}
Warning: Tap point (100.0, 200.0) on layer 3 for net VDD does not touch any polygon (RAIL-305)
```

The following example creates a tap using the same absolute coordinates but disables the checking. No warning message is issued. Is_valid status is shown as unknown.

```
pr_shell> create_taps -supply VDD -layer 3 -point {100.0 200.0} -nocheck
1
pr_shell> get_attribute [get_tap TAP1] is_valid
unknown
```

The following example creates the 101th tap point using the same absolute coordinates. No warning message is issued since only the top 100 taps are to be checked.

```
pr_shell> create_taps -supply VDD -layer 3 -point {100.0 200.0}
```

The following example sets the *rail_force_tap_check* variable to true and creates the 101th tap point using the same absolute coordinates. A warning message is issued since it does not touch any layout geometry.

```
pr_shell> set rail_force_tap_check true
pr_shell> create_taps -supply VDD -layer 3 -point {100.0 200.0}
Warning: Tap point (100.0, 200.0) on layer 3 for net VDD does not touch any polygon (RAIL-305)
```

The following example imports taps from the rail context into the session context, assuming the rail context consists of taps rail_tap1 and rail_tap2.

```
pr_shell> remove_taps
pr_shell> create_taps -result rail
pr_shell> get_taps
{"rail_tap1", "rail_tap2"}
pr_shell> report_taps

Name      Type       Net    Point             Layer        Object Name     Context      Is_valid
-----------------------------------------------------------------------------
rail_tap1 auto_pg    VDD    (100.000 200.000) (3) M2       /MyTop/VDD      session      true
rail_tap2 auto_pg    VSS    (300.230 302.180) (1) M1       /MyTop/VSS      session      true
```

The following example imports taps from a PrimeRail tap file.

```
pr_shell> exec cat tap_file
VDD 3 500.000 500.000
VSS 5 500.000 500.000
pr_shell> create_taps -import tap_file
```

The following example imports taps from a PrimeRail tap file with SPICE package connection. See also **set_tap_package_model**

```
pr_shell> exec cat tap_file
VDD 3 500.000 500.000 PKG_PORT PKG_VDD
VSS 5 500.000 500.000 PKG_PORT PKG_VSS
pr_shell> create_taps -import tap_file
```

The following example imports taps from an IC Compiler virtual power pad file.

```
pr_shell> exec cat pad_file
VDD
555.000 555.000 3
666.000 666.000 2
VSS
555.000 555.000 5
pr_shell> create_taps -import pad_file
```

The following example snaps a tap from the original location to the nearest object using 100 micron as the snap distance.

```
pr_shell> create_taps -point {50 50} -layer M1 -snap_distance 100
Snap tap point from original location (50.000, 50.000) to new location (90.000, 65.910)
1
```

SEE ALSO

```
check_taps(2)
current_integrity_result(2)
current_rail_result(2)
get_taps(2)
rail_attributes(2)
rail_force_tap_check(3)
remove_taps(2)
report_integrity_result(2)
report_rail_result(2)
report_taps(2)
set_tap_package_model(2)
tap_contexts(2)
write_taps(2)
```

create_voltage_area

Creates a voltage area at the specified region for providing placement constraints of cells associated with the region.

SYNTAX

```
status create_voltage_area
   [-name voltage_area_name]
   [-coordinate coordinate_list]
   [-polygons polygon_coordinate_list]
   [-power_domain power_domain_name]
   [-guard_band_x guard_band_width]
   [-guard_band_y guard_band_width]
   cell_list
```

Data Types

voltage_area_name	string
coordinate_list	list
polygon_coordinate_list	list
power_domain_name	string
guard_band_width	integer
cell_list	list

ARGUMENTS

-name *voltage_area_name*

Specifies the name of the voltage area to be created. If there is a voltage area with the same name already, the voltage area cannot be created. The name *DEFAULT_VA* is reserved and cannot be used for a user-defined voltage area. The **-name** and **-power_domain** options are mutually exclusive.

-coordinate *coordinate_list*

Specifies a list of lower-left and upper-right coordinates of a voltage area. Specify the *coordinate_list* in the format of {llx1 lly1 urx1 ury1 …}, where each set of {llx lly urx ury} defines a target rectangular placement area. The voltage area geometry can be a rectangle or a rectilinear polygon. The unit is micron. The **-coordinate** and **-polygons** options are mutually exclusive.

-polygons *polygon_coordinate_list*

Specifies a list of polygons of a voltage area. Specify the *polygon_coordinate_list* in the format of {{{x1 y1} … {xN yN} {x1 y1}} …}, where each set of {x1 y1 … xN yN x1 y1} defines the list of points of one polygon. The voltage area geometry can be a rectangle (five points) or a rectilinear polygon. The unit is micron. The **-coordinate** and **-polygons** options are mutually exclusive.

-power_domain *power_domain_name*

Creates a voltage area from an existing power domain. The name of the voltage area is identical to the name of the power domain. All hierarchical cells associated with the power domain are included in the voltage area. After the voltage area is successfully created, the power domain contains the corresponding boundary information. The **-name** and **-power_domain** options are mutually exclusive.

-guard_band_x ***guard_band_width***

Specifies the guard width in the horizontal direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive integer.

-guard_band_y ***guard_band_width***

Specifies the guard width in the vertical direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive integer.

cell_list

Specifies a list of hierarchical cells that are contained in the voltage area. Multiple hierarchies are permitted, and one cell cannot be in two voltage areas. Leaf cells (or standard cells) cannot be specified in this option.

You must specify cell_list option when you specify the **-name** option, unless you use the **-power_domain** option.

DESCRIPTION

The **create_voltage_area** command creates a voltage area of specific geometry on the core area of the chip. The voltage area is associated with hierarchical cells. PrimeTime uses the voltage area information to insert ECO cells in correct locations if you use the **fix_eco_drc** or **fix_eco_timing** command with the **-physical_mode** option.

You can create a voltage area only after PrimeTime loads the physical database. To specify a file that contains **create_voltage_area** commands, use the **set_eco_options** command with the **-physical_constraint_file** option. This enables PrimeTime to load physical database, read the constraint file, and create voltage areas before starting to fix violations. The **create_voltage_area** commands in **read_sdc** are ignored.

Voltage areas can physically be completely nested; that is, one voltage area can lie completely inside another voltage area.

Voltage areas can also be logically nested; that is, one hierarchical cell can belongs to one voltage area, while one of its descendants belongs to another voltage area.

Voltage areas cannot be partially overlapped. The creation fails if you try to create a voltage area partially overlapping with an existing voltage area.

EXAMPLES

The following example creates a voltage area named V_AREA1 for hierarchical cell INST_1.

```
pt_shell> create_voltage_area -name V_AREA1 \
```

```
          -coordinate {100 100 200 200} INST_1
```

The following example creates a voltage area from an existing power INST. domain:

```
pt_shell> create_voltage_area -power_domain INST \
          -coordinate { 215 215 350 350 }
```

The following example creates a voltage area by using the **-polygons** option:

```
pt_shell> create_voltage_area -power_domain INST \
          -polygons {{100 100} {300 100} {300 200} {200 200} {200 300} \
          {100 300} {100 100}}
```

The following example creates a voltage area with disjoint polygons by using the **-polygons** option:

```
pt_shell> create_voltage_area -power_domain INST \
          -polygons {{{100 100} {300 100} {300 200} {200 200} {200 300} \
          {100 300} {100 100}} {{400 400} {500 400} {500 500} {400 500} \
          {400 400}}}
```

SEE ALSO

```
create_placement_blockage(2)
fix_eco_drc(2)
fix_eco_timing(2)
set_eco_options(2)
write_changes(2)
```

current_design

Sets or returns the current design.

SYNTAX

```
string current_design
   [design_name]
```

Data Types

```
design_name     string
```

ARGUMENTS

design_name

Specifies the working or focal design for many PrimeTime commands. If you omit the *design_name* option, the **current_design** command returns a collection containing the current design. If the *design_name* option specifies a design that cannot be found, an error is issued and the working design remains unchanged.

DESCRIPTION

This command sets or returns the working design, which is the focus of many PrimeTime commands. Before **link_design current_design** won't return anything. Without arguments, the **current_design** command returns a collection containing the current working design. Together, the **current_design** and **current_instance** commands define the focus for many PrimeTime commands.

To display designs currently available in PrimeTime, use the **list_designs** command.

EXAMPLES

The following example uses the **current_design** command to show the current context and to change the context from one design to another.

```
pt_shell> current_design
{"TOP"}

pt_shell> list_designs
```

```
Design Registry:
    ADDER             /designs/dbs/my_design.db:ADDER
    FULL_ADDER        /designs/dbs/my_design.db:FULL_ADDER
    FULL_SUBTRACTOR   /designs/dbs/my_design.db:FULL_SUBTRACTOR
    HALF_ADDER        /designs/dbs/my_design.db:HALF_ADDER
    HALF_SUBTRACTOR   /designs/dbs/my_design.db:HALF_SUBTRACTOR
    SUBTRACTOR        /designs/dbs/my_design.db:SUBTRACTOR
 *  TOP               /designs/dbs/my_design.db:TOP

pt_shell> current_design ADDER
{"ADDER"}

pt_shell> current_design
{"ADDER"}
```

The **current_design** command can be used as a parameter to other **pt_shell** commands. In the following example, the **remove_design** command is used to delete the working design from **pt_shell**.

```
pt_shell> current_design
{"TOP"}
pt_shell> remove_design [current_design]
Removing design 'TOP'...
1
pt_shell> current_design
Error: Current design is not defined. (DES-001)
pt_shell>
```

SEE ALSO

current_instance(2)
list_designs(2)

current_instance

Sets the working instance object and enables other commands to be used relative to a specific instance in the design hierarchy.

SYNTAX

```
string current_instance
   [instance]
```

Data Types

```
instance     string
```

ARGUMENTS

instance

Specifies the working instance (cell), which is the command focus.

DESCRIPTION

The **current_instance** command specifies the working instance. An instance is a cell in the hierarchy of a design. This command differs from the **current_design** command, which changes the working design and then sets the current instance to the top level of the new current design. The combination of the current design and current instance defines the focus for many PrimeTime commands.

The **current_instance** command traverses the design hierarchy similar to the way the UNIX **cd** command traverses the file hierarchy. The **current_instance** command operates with a variety of *instance* arguments:

- If you do not specify an *instance* argument, the command focus returns to the top level of the hierarchy.
- If *instance* is one period (.), the command returns the current instance, and no change is made.
- If *instance* is two periods (..), the current instance moves up one level in the design hierarchy.
- If *instance* is a valid cell (or cell name) at the current level of hierarchy, the command focus moves down to that level of the design hierarchy.

EXAMPLES

The following example uses **current_instance** to move up and down the design hierarchy. The **all_instances** command is also used to list instances of a specific design relative to the current instance.

```
pt_shell> current_design TOP
{"TOP"}

pt_shell> current_instance U1
U1

pt_shell> current_instance "."
U1

pt_shell> query_objects [all_instances ADDER]
{"U1", "U2", "U3", "U4"}

pt_shell> current_instance U3
U1/U3

pt_shell> current_instance "../U4"
U1/U4

pt_shell> current_instance .
Current instance is the top-level of design 'TOP'.
```

The following example shows how to descend multiple levels of hierarchy by separating cell names with a slash:

```
pt_shell> current_instance U1/A2/B3
```

The following command moves the command focus up two levels of hierarchy, then down one level to the MY_INST cell.

```
pt_shell> current_instance "../../MY_INST"
```

In the following example, changing the **current_design** resets the **current_instance** to the top level of the new design hierarchy.

```
pt_shell> current_design
{"TOP"}

pt_shell> current_instance "U2/U1"
U2/U1

pt_shell> current_design ADDER
{"ADDER"}

pt_shell> current_instance
Current instance is the top-level of design 'ADDER'.
```

The following example uses **current_instance** to go to an instance of another design. The current_design is set by the new design whose name is the name after the first slash of the given instance name.

```
pt_shell> current_design
{"TOP"}

pt_shell> current_instance U1
U1

pt_shell> current_instance "/TOP/U2"
U2

pt_shell> current_instance "/ALARM_BLOCK/U6"
U6
```

```
pt_shell> current_design
{"ALARM_BLOCK"}
```

SEE ALSO

all_instances(2)
current_design(2)
list_designs(2)
query_objects(2)

current_integrity_result

Sets or deletes the current integrity result.

SYNTAX

```
string current_integrity_result
   [result_name]
   [-clear]
```

Data Types

```
string result_name
```

ARGUMENTS

result_name

Specifies the name of an existing integrity result to be assigned as current in the session.

-clear

Deletes the current integrity result from memory without replacing it with a different result, so that there is no integrity result loaded.

DESCRIPTION

The command sets or deletes the current integrity result and returns the name of the integrity result that is now current. If no parameters or options are specified, the name of the current integrity result is returned. Otherwise, the command returns none if there is no current integrity result.

Whenever the **check_supply_net_integrity** command is successfully run, a named integrity result is automatically created. The result contains the data generated by the command and a snapshot of all taps created in the session (see **tap_contexts** for more detailed descriptions on tap contexts.) When you review the supply net integrity problems in the error browser, the data that is used to provide the error information comes from the current integrity result. When new results are created by executing additional **check_supply_net_integrity** commands, the new results replace the existing results so that the error browser uses the latest results. Previous results can be reloaded by using the **current_integrity_result** command to specify a previous result as current. This allows you to toggle between two or more result sets. The **report_integrity_results** command is used to see which integrity results already exist in the current session for subsequent toggling.

Use the -clear option to clear the current integrity result without replacing it with a new integrity result, as if the **check_supply_net_integrity** command had not been executed. The -clear option and the *result_name*

parameter are mutually exclusive. Note that if you use the -clear option, the result is still available in the session, so it can be reloaded later with the **current_integrity_result** command. Use the **remove_integrity_result** command to completely remove the integrity result from the current session.

The integrity checking errors that are displayed in the GUI error browser are from the currently loaded integrity result. If the **current_integrity_result** command is used to change the integrity result while the error browser is displayed, the error browser is updated to reflect the currently selected result.

Integrity result data is cached to disk when it is created. The disk files are stored within the *pr_work_dir* tree, and is managed exclusively by the tool. The result files should not be altered by the user. If the result files are corrupted, the behavior of the tool is undefined.

EXAMPLES

The following example switches between different integrity results. For this example, there are two runs of **check_supply_net_integrity**, under different setup conditions, creating two different integrity result sets.

```
pr_shell> check_supply_net_integrity -result MyResult_1 -floating
...
1
pr_shell> check_supply_net_integrity -result MyResult_2 -missing_via_levels 2
...
1
pr_shell> current_integrity_result
MyResult_2
pr_shell> current_integrity_result MyResult_1
MyResult_1
pr_shell> current_integrity_result
MyResult_1

The following example removes the current integrity result
without replacing it with another integrity result.

pr_shell> current_integrity_result MyResult_2
MyResult_2
pr_shell> current_integrity_result -clear
none
pr_shell> current_integrity_result
none
```

SEE ALSO

```
check_supply_net_integrity(2)
remove_integrity_result(2)
report_integrity_results(2)
pr_work_dir(3)
```

current_mw_cel

Sets the working Milkyway design in the tool or reports the current design.

SYNTAX

```
collection current_mw_cel
   [mw_cel]
```

Data Types

mw_cel	cel

ARGUMENTS

mw_cel

Sets the specified a Milkyway design to be the current working Milkyway design. If no name is provided, the command returns the name of the current working Milkyway design.

You can specify a Milkyway design by name, name pattern, or the Milkyway design collection's name. For example, *top* matches a Milkyway design named *top* in the current library and *top** matches all Milkyway designs that have names beginning with *top*. However, a name pattern must match only one existing Milkyway design because only one design can be set as the current design.

DESCRIPTION

This command sets or reports the working Milkyway design.

If you do not specify a *mw_cel* value, the command returns the name of the current Milkyway design.

If you specify an *mw_cel* value, the command first determines whether the specified Milkyway design is an open Milkyway design. If the Milkyway design is not open, it instructs you to open the Milkyway design first. If it is an open Milkyway design, the command sets the current Milkyway design to that design.

The *mw_cel* option must be a top design. If the top design specified has a stack, the stack is cleared, which means that all the child designs in the stack are popped out. For more information regarding the design stack, see man page for the **change_working_design** command. To switch to a design stack without clearing it, see the man page for the **change_working_design_stack** command.

The following three commands change the value of the current Milkyway design:

- The **open_mw_cel** command automatically sets the opened Milkyway design as the current Milkyway design.

- The **create_mw_cel** command automatically sets the newly created Milkyway design as the current Milkyway design.
- The **close_mw_cel** command pops closed Milkyway designs from all stacks and switches the current Milkyway design to the top of the most recently used stack.
- The **close_mw_lib** command clears the current Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples use the **current_mw_cel** command to show the current context and to change the context from one Milkyway design to another.

```
prompt> current_mw_cel
{"top"}

prompt> current_mw_cel [get_mw_cel ADDER]
{"ADDER"}

prompt> current_mw_cel
{"ADDER"}
```

The following examples use the **close_mw_cel** command to close the working Milkyway design in the tool:

```
prompt> current_mw_cel
{"TOP"}

prompt> close_mw_cel
1

prompt> current_mw_cel
Error: No mw cel is open
```

SEE ALSO

```
close_mw_cel(2)
copy_mw_cel(2)
create_mw_cel(2)
get_mw_cels(2)
mw_cel_collection(2)
open_mw_cel(2)
remove_mw_cel(2)
rename_mw_cel(2)
save_mw_cel(2)
get_working_design_stack(2)
change_working_design_stack(2)
change_working_design(2)
```

current_mw_lib

Gets the current Milkyway library.

SYNTAX

```
collection current_mw_lib
```

ARGUMENTS

The **current_mw_lib** command has no arguments.

DESCRIPTION

The **current_mw_lib** command gets the current Milkyway library. The **current_mw_lib** command returns a collection of the current Milkyway library, if one exists.

Many Milkyway library-related commands use the current Milkyway library by default. The current Milkyway library is automatically set when the **open_mw_lib** command opens a Milkyway library to use.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the current Milkyway library in the current session:

```
prompt> current_mw_lib
{"design"}
```

SEE ALSO

close_mw_lib(2)
copy_mw_lib(2)
create_mw_lib(2)

```
open_mw_lib(2)
rebuild_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)
```

current_rail_result

Sets or deletes the current rail result.

SYNTAX

```
string current_rail_result
   [name]
   [-clear]
```

Data Types

```
string name
```

ARGUMENTS

name

Specifies the name of an existing rail result to be assigned as current.

-clear

Deletes the current rail result from memory without replacing it with a different result.

DESCRIPTION

The command sets or clears the current rail result and returns the name of the rail result that is now current. If no parameters or options are specified, the name of the current rail result is returned. The command returns none if there is no current rail result.

Whenever a rail analysis calculation command, such as **calculate_rail_voltage** or **calculate_minimum_path_resistances**, is successfully run, a named rail result is automatically created by that command. The result file contains the data produced by the analysis and a snapshot of taps that are created during the command run (see **tap_contexts** for a more detailed description on tap contexts). When you query the *voltage_drop* attribute of an instance's PG pin or examine a voltage drop map in the GUI, the data that is used to provide this information comes from the current rail result. As new results are created by executing an additional **calculate_rail_voltage** command, the new result replaces the existing results so that any later queries or map displays are done with respect to these new results. To reload and assign a previous result as current, run the **current_rail_result** command. This allows you to toggle between two or more result sets. Use the **report_rail_results** command to see which rail results already exist in the current session for subsequent toggling.

Set the -clear option to delete the current rail result without replacing it with a new rail result, as if an analysis

command had not been executed. All taps in the rail context will also be cleared. The -clear option and the *result_name* parameter are mutually exclusive.

Note that using the -clear option does not remove the result from the session, and you can reload and assign it as current later with the **current_rail_result** command. Use the **remove_rail_result** command to completely remove the rail result from the current session.

The rail maps that are displayed in the GUI are from the currently loaded rail result. If the **current_rail_result** command is used to change the rail result while the map is displayed in the GUI, the tool automatically updates the map display to reflect the change.

The tool manages the rail result data in both memory and disk for caching and memory management reasons but not for direct user access. If this data is corrupted in the disk cache, then behavior is undefined. The data is cached in the application's session working directory. See the **pr_work_dir** man page for further details.

EXAMPLES

The following example switches between different rail result sets. For this example, there are two runs of the **calculate_rail_voltage** command, under different setup conditions. Two named rail results are generated, including the voltage drop analysis result of the same net.

```
pr_shell> calculate_rail_voltage -static -result MyResult_1 VDD
...
1
pr_shell> create_tap -of [get_cell MyTop/U235]
1
pr_shell> calculate_rail_voltage -static -result MyResult_2 VDD
...
1
pr_shell> current_rail_result
MyResult_2
pr_shell> get_attribute [get_pg_pin U1/VDD] static_voltage
-2.14354e-04
pr_shell> current_rail_result MyResult_1
MyResult_1
pr_shell>  get_attribute [get_pg_pin U1/VDD] static_voltage
-1.41374e-05

The following example removes the current rail result without
replacing it with another rail result.

pr_shell> current_rail_result myrailresult
myrailresult
pr_shell> get_attribute [get_pg_pin U123/VDD] static_voltage
-3.19933e-05
pr_shell> current_rail_result -clear
none
pr_shell> current_rail_result
none
pr_shell> get_attribute [get_pg_pin U123/VDD] static_voltage
UNINIT
```

SEE ALSO

```
calculate_minimum_path_resistances(2)
calculate_rail_voltage(2)
remove_rail_result(2)
rename_rail_result(2)
report_rail_results(2)
pr_work_dir(3)
```

date

Returns a string containing the current date and time.

SYNTAX

```
string date
```

DESCRIPTION

The **date** command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

```
ddd mmm nn hh:mm:ss yyyy
```

Where:

```
ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year
```

The **date** command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with **exec** to ask for the date and time.

EXAMPLES

The following command prints the date.

```
prompt> echo "Date and time: [date]"
Date and time: Thu Dec  9 17:29:51 1999
```

SEE ALSO

exec(2)

define_design_mode_group

Defines a design mode group with a set of design modes.

SYNTAX

```
string define_design_mode_group
   [-group_name name]
   mode_list
```

Data Types

```
name          string
mode_list     list
```

ARGUMENTS

-group_name *name*

Specifies the name of the design mode group. By default, if no design mode group is specified, a default design mode group is created and named "default".

mode_list

Specifies a list of design modes to be created and included in the design mode group.

DESCRIPTION

The **define_design_mode_group** command defines a set of design modes to be placed in a design mode group for your design. You select one of the design modes to be the active mode using the **set_mode -type design** command. There are only two valid states for a design mode group. The default state of all modes enabled or the state of one mode enabled and all others disabled. Setting one of the modes to be active automatically sets the rest of the modes in the group inactive (disabled).

You can map cell modes or paths to a design mode using the **map_design_mode** command. When you select the active design mode with the **set_mode -type design** command, the modes of the cells specified for that design mode are also active. Any paths mapped to disabled design modes automatically become false paths.

Unlike design modes, which are user-specified, cell modes are predefined, and are in the library. You can find out what cell modes are available using the **report_mode** command. For more information about cell modes, see the manual page for the **set_mode** command.

To see a report of the design modes specified for the current design, use the **report_mode -type design** command.

To remove design modes, use the **remove_design_mode** command. Removing a design mode also removes any cell mode or path specifications previously mapped to the design mode using the **map_design_mode** command.

EXAMPLES

The following example defines two design modes for the current design: SYSTEM and TEST.

```
pt_shell> define_design_mode_group {SYSTEM TEST}
```

The following example defines two independent design mode groups. The first command defines a design mode group named RW that contains design modes READ and WRITE. The second command defines a design mode group named LT that contains design modes LATCH and TRANSPARENT.

```
pt_shell> define_design_mode_group -group RW {READ WRITE}
pt_shell> define_design_mode_group -group LT {LATCH TRANSPARENT}
```

The following command enables the design mode READ and disables the design mode WRITE, because READ and WRITE belong to the same design mode group RW.

```
pt_shell> set_mode -type design READ
```

SEE ALSO

remove_design_mode(2)
map_design_mode(2)
report_mode(2)
reset_mode(2)
set_mode(2)

define_name_maps

Defines object name mapping for an application and design.

SYNTAX

```
status define_name_maps
   -application application_name
   -design design_name
   -columns column_list
   entries
```

Data Types

```
application_name   string
design_name        string
column_list        list
entries            list
```

ARGUMENTS

-application *application_name*

Specifies the targeting application name. Currently, only **golden_upf** is supported. The **-application** and **-design** options jointly specify a unique in-memory name map instance for the current session.

-design *design_name*

Specifies the name of the design. The **-application** and **-design** options jointly specify a unique in-memory name map instance for the current session.

-columns *column_list*

Specifies the column names of the name map for the *application_name*.

entries

Specifies name mapping entries conforming to the definition of *column_list*.

DESCRIPTION

The **define_name_maps** command specifies a list of object name mapping entries for a specified application. Usually, a name map is associated with a particular design name. *application_name* is predefined by the tool.

The object name map could help in certain constraint applications where the default object query or even the rule-

based query algorithms could not succeed in finding the correct objects. The typical case is when trying to apply RTL constraints onto the postoptimization netlist.

Note: It is the specific application that defines the content format and the usage model of its name map. For details about the functionality of a name map, see the documentation for the specific application.

APPLICATION golden_upf

golden_upf is the application_name assigned to golden UPF flow. Command **define_name_maps** for **golden_upf** could appear in golden UPF name mapping file, although other command such as **set_query_rules** could also exist in the golden UPF name mapping file. **golden_upf** name map has four columns. They are **class**, **pattern**, **options** and **names**.

The **class**, **pattern** and **options** columns are used jointly to locate the entry in the map; the **names** column is the mapping result.

The following example illustrates the map contents and columns.

```
prompt> define_name_maps                      \
    -application golden_upf                   \
    -design TOP                               \
    -columns {class pattern options names}    \
    {cell mid/inst/U3     {} {mid_inst/U4 mid_inst/U5}} \
    {cell mid/inst/bot+U5 {} {mid_inst/bot/U9}}
```

The preceding name map is for application **golden_upf** and the TOP design. The following four columns are predefined for the application:

- **class** specifies an object class name such as cell or port. The supported class names are cell, port, pin, net, power_domain, power_switch, supply_net, supply_port and supply_set.
- **pattern** specifies object matching pattern string. This is one of the object name pattern from the golden UPF file. The optional delimiter character '+' is used to indicate that a pattern is composed of a matching string (for example, "U5") and a scope (for example, scope "mid/inst/bot") for that matching string.
- **options** specifies a list of matching options for the map entry. It is derived from the golden UPF file. The supported **options** is a list of the following elements {**nocase transitive leaf**|**noleaf in**|**out**|**inout**}. **nocase** option specifies that the string in the **pattern** column should be treated case-insensitively during name lookups. For the meanings of other options, see the man page of the **find_objects** command. **leaf** and **noleaf** are mutually exclusive. **in**, **out** and **inout** are mutually exclusive. In the preceding example, no option is used.
- **names** is a list object names where every name element is a full-path name relative to the top scope of the specified design. Column **names** is the name mapping target.

The preceding name map would result in the following query behavior under the **golden_upf** application, although in real golden UPF flow, the queries are executed through **load_upf** command:

```
prompt> find_objects . -pattern mid/inst/U3 -object_type inst
{mid_inst/U4 mid_inst/U5}

prompt> find_objects mid/inst/bot -pattern U5 -object_type inst
{mid_inst/bot/U9}
```

SEE ALSO

find_objects(2)

```
load_upf(2)
set_query_rules(2)
```

define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

SYNTAX

```
string define_proc_attributes
   proc_name
   [-info info_text]
   [-define_args arg_defs]
   [-define_arg_groups group_defs]
   [-command_group group_name]
   [-return type_name]
   [-hide_body]
   [-hidden]
   [-dont_abbrev]
   [-permanent]
```

Data Types

proc_name	string
info_text	string
arg_defs	list
group_defs	list
group_name	string
type_name	string

ARGUMENTS

proc_name

Specifies the name of the existing procedure.

-info *info_text*

Provides a help string for the procedure. This is printed by the **help** command when you request help for the procedure. If you do not specify *info_text*, the default is "Procedure".

-define_args *arg_defs*

Defines each possible procedure argument for use with **help -verbose**. This is a list of lists where each list element defines one argument.

-define_arg_groups *group_defs*

Defines argument checking groups. These groups are checked for you in parse_proc_arguments. each list element defines one group

-command_group ***group_name***

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-return \ftype_name

Specifies the type of value returned by this proc. Any value may be specified. Some applications use this information for automatically generated dialogs setc. Please see the type_name option attribute described below.

-hide_body

Hides the body of the procedure from **info body**.

-hidden

Hides the procedure from **help** and **info proc**.

-dont_abbrev

Specifies that the procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the **sh_command_abbrev_mode** variable.

-permanent

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

DESCRIPTION

The **define_proc_attributes** command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the **proc** command, it is placed in the Procedures command group. There is no help text for its arguments. You can view the body of the procedure with **info body**, and you can modify the procedure and its attributes. The **define_proc_attributes** command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named, positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The **define_proc_attributes** command does not relate the information that you enter for argument definitions with **-define_args** to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using **parse_proc_arguments** to validate and extract your arguments.

The *info_text* is displayed when you use the **help** command on the procedure.

Use **-define_args** to define help text and constraints for individual arguments. This makes the help text for the procedure look like the help for an application command. The value for **-define_args** is a list of lists. Each element has the following format:

```
arg_name option_help value_help data_type attributes
```

The elements specify the following information:

- *arg_name* is the name of the argument.

- *option_help* is a short description of the argument.
- *value_help* is the argument name for positional arguments, or a one word description for dash options. It has no meaning for a Boolean option.
- *data_type* is optional and is used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string.
- *attributes* is optional and is used for option validation. The *attributes* is a list that can have any of the following entries:
  - "required" - This argument must be specified. This attribute is mutually exclusive with optional.
  - "optional" - Specifying this argument is optional. This attribute is mutually exclusive with "required."
  - "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.
  - "values {<list of allowable values>}" - If the argument type is one_of_string, you must specify the "values" attribute.
  - "type_name <name>" - Give a descriptive name to the type that this argument supports. Some applications may use this information to provide features for automatically generated dialogs, etc. Please see product documentation for details. This attribute is not supported on boolean options.
  - "merge_duplicates" - When this option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.
  - "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.

  The default for *attributes* is "required."

Use the **-define_arg_groups** to define argument checking groups. The format of this option is a list where each element in the list defines an option group. Each element has the following format:

{<type> {<opt1> <opt2> ...} [<attributes>]}

Currently two types of groups are supported:

- *"exclusive"* Only one option in an exclusive group is allowed. All the options in the group must have the same required/optional status. This group can contain any number of options.
- *"together"* If the first option in the group is specified then the second argument is also required. This type of group can contain at most two options.
- \fi"related" These options are related to each other. An automatic dialog builder for this command may try to group these options together.

The supported attributes are:

- *{"label" <text>}* An optional label text to identify this group. The label may be used by an application to automatically build a grouping in a generated dialog.
- *"bidirectional"* This is only valid for a together group. It means that both options must be specified together.

Change the command group of the procedure using the **-command_group** command. Protect the contents of the procedure from being viewed by using **-hide_body**. Prevent further modifications to the procedure by using **-permanent**. Prevent abbreviation of the procedure by using **-dont_abbrev**.

EXAMPLES

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} { return [expr $a + $b]}

prompt> define_proc_attributes plus -info "Add two numbers" \
? -define_args {
  {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}

prompt> help -verbose plus
Usage: plus    # Add two numbers
    [-verbose]         (issue a message)
    a                  (first addend)
    b                  (second addend)

prompt> plus 5 6
11
```

In the following example, the argHandler procedure accepts an optional argument of each type supported by **define_proc_attributes**, then displays the options and values received. Note the only one of -Int, -Float, or -Bool may be specified to the command:

```
proc argHandler {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}

define_proc_attributes argHandler \
  -info "Arguments processor" \
  -define_args {
     {-Oos "oos help"       AnOos   one_of_string
        {required value_help {values {a b}}}}
     {-Int "int help"       AnInt   int     optional}
     {-Float "float help"   AFloat  float   optional}
     {-Bool "bool help"     ""      boolean optional}
     {-String "string help" AString string  optional}
     {-List "list help"     AList   list    optional}
     {-IDup "int dup help"  AIDup   int     {optional merge_duplicates}}
   } \
  -define_arg_groups {
     {exclusive {-Int -Float -Bool}}
   }
```

SEE ALSO

```
help(2)
info(2)
parse_proc_arguments(2)
proc(2)
```

sh_command_abbrev_mode(3)

define_qtm_attribute

Defines a new user-defined attribute for a class of Quick Timing Model (QTM) objects.

SYNTAX

```
string define_qtm_attribute
   -type data_type
   -class obj_class
   attr_name
```

Data Types

```
data_type          string
obj_class          string
attr_name          string
```

ARGUMENTS

-type *data_type*

Specifies the data type of the attribute. The supported data types are string, int, float, and Boolean.

-class *obj_class*

Specifies the class of QTM objects the new attribute is defined for. The valid object classes are lib, lib_cell or lib_pin. Attribute for 'lib' class applies to the library. Attribute for 'lib_cell' class applies to the QTM cell. Attribute for 'lib_pin' class applies to the QTM ports/pins.

attr_name

Specifies the name of the attribute.

DESCRIPTION

Use **define_qtm_attribute** command to define a new user-defined attribute that is applicable to QTM objects. A user-defined attribute is any attribute which PrimeTime does not understand by default. Those attributes already understood or reserved by PrimeTime for various classes of objects are considered application attributes and should not be re-defined. If you need to set the application attributes that are applicable to the classes of QTM objects, just use **set_qtm_attribute** directly. There is no need to define the attribute before setting them. But for user attributes that are not reserved by the application, you have to define them before set them on objects. Note that the **list_attributes** command will not show any attributes defined for QTM.

After save the QTM in proper library, those user-defined QTM attributes can be imported from db files. But you

must define those user attributes you want to import with the **define_user_attribute** command and the *-import* option before the library with the QTM cell is read.

For more information, see the **define_user_attribute** man page.

EXAMPLES

The following example defines QTM attributes of various types.

```
pt_shell> define_qtm_attribute my_str_attr \
           -class lib_pin -type string
pt_shell> define_qtm_attribute my_int_attr \
           -class lib_cell -type int
pt_shell> define_qtm_attribute my_float_attr \
           -class lib -type float
```

SEE ALSO

```
set_qtm_attribute(2)
remove_qtm_attribute(2)
define_user_attribute(2)
```

define_scaling_lib_group

Defines a group of libraries for voltage and temperature scaling.

SYNTAX

```
string define_scaling_lib_group
   [-exact_match_only]
   library_list
```

Data Types

```
library_list     list
```

ARGUMENTS

-exact_match_only

Specifies that this scaling group is analyzed in exact-match only mode. The tool skips scaling and tries to find the exact-match library. If not found, the linked library is used, and the SLG-216 warning is issued.

library_list

Specifies a list of libraries for voltage and temperature scaling. Only one of the libraries in the *library_list* option should be in the **link_path** variable; the remaining libraries are read automatically after the design is linked to complete the group.

A minimum of two libraries is required for one-dimensional scaling (voltage or temperature), and a minimum of four libraries is required for two-dimensional scaling (voltage and temperature).

DESCRIPTION

The **define_scaling_lib_group** command defines a group of libraries that PrimeTime interpolates between for voltage and temperature scaling. Scaling with library groups supports timing, noise, and power analysis using Composite Current Source (CCS) model data, nonlinear delay model (NLDM) data, or a mix of both.

Read and link the design before using the **define_scaling_lib_group** command. If the design is not already linked, this command automatically links the design.

You can have more than one group to cover different portions of a design, but the groups cannot share libraries.

When a group is in effect, you cannot use the **set_rail_voltage** or **set_operating_conditions** command outside the range of the applicable group.

To see if a group is used for delay calculation, use the **report_delay_calculation** command. The report shows the results using the link library, then the scaling library group results if applicable and valid.

To report all scaling groups defined by the **define_scaling_lib_group** command, use the **report_lib_groups** command.

EXAMPLES

The following example defines a scaling library group for two voltage domains about the nominal:

```
pt_shell> set link_path "* 1.1.db"
pt_shell> link_design
pt_shell> define_scaling_lib_group { 0.9.db 1.1.db 1.3.db }
```

SEE ALSO

```
link_design(2)
report_delay_calculation(2)
report_lib_groups(2)
set_operating_conditions(2)
set_rail_voltage(2)
```

define_user_attribute

Defines a new user-defined attribute.

SYNTAX

```
string define_user_attribute
   -type string | int | float | double | boolean
   -classes class_list
   [-range_min min]
   [-range_max max]
   [-one_of values]
   [-import]
   [-quiet]
   attr_name
```

Data Types

class_list	list
min	double
max	double
values	list
attr_name	string

ARGUMENTS

-type string | int | float | double | boolean

Specifies the data type of the attribute.

-classes *class_list*

Defines the attribute for one or more of the classes. The valid object classes are **design**, **port**, **cell**, **pin**, **net**, **lib**, **lib_cell**, or **lib_pin**.

-range_min *min*

Specifies *min* value for numeric ranges. This is only valid when the data type is **int** or **double**. Specifying a minimum constraint without a maximum constraint creates an attribute which accepts a value >= *min*.

-range_max *max*

Specifies *max* value for numeric ranges. This is only valid when the data type is **int** or **double**. Specifying a maximum constraint without a minimum constraint creates an attribute which accepts a value <= *max*.

-one_of *values*

Provides a list of allowable strings. This is only valid when the data type is **string**.

-import

Imports this attribute from a design or library database.

-quiet

Does not report any messages.

attr_name

Specifies the name of the attribute.

DESCRIPTION

Use the **define_user_attribute** command to define a new user-defined attribute. A user-defined attribute is any attribute which PrimeTime does not understand by default. The **list_attributes** command shows all attributes which PrimeTime understands.

You can apply attributes to most object classes in PrimeTime. You can use these to mark interesting cells or nets, or store values you computed, and so on. PrimeTime cannot use these attributes, but you can use them in scripts, procedures, and so on. You can list the attributes you defined using the **list_attributes** command.

User-defined attributes can be imported from a design or library database after they have been defined. A design or library database is a db or ddc format file, or a Milkyway database. You must define attributes you want to import with the **-import** option. Note that imported attributes do not appear on the design objects until the design is linked. Attributes imported from a design or library database are inherited through the hierarchy. Note that attributes defined for designs are inherited onto cells, and attributes defined for ports are inherited onto pins. The **define_user_attribute** command sets up these relationships for you automatically if you do not do it explicitly.

EXAMPLES

The following example defines attributes of various types. Note that more than one class can be specified. Also note that attr_i is defined as imported from a design or library database. Finally, attribute design1 is imported from a design or library database, and PrimeTime defines it for cells as well.

```
pt_shell> define_user_attribute attr_s \
            -classes {cell net} -type string
pt_shell> define_user_attribute attr_i \
            -classes cell -type int -import
pt_shell> define_user_attribute design1 \
            -classes design -type int -import
Information: Inferred definition of attribute 'design1' for class 'cell'
        because it is imported for class 'design' (ATTR-7)
```

In the following example, attr_ir1 is defined as >= 0 and <= 100, attr_ir2 is defined as >= 0 with no maximum, and attr_ir3 is defined as <= 100 with no minimum.

```
pt_shell> define_user_attribute attr_ir1 \
            -classes cell -type int \
            -range_min 0 -range_max 100
pt_shell> define_user_attribute attr_ir2 \
            -classes cell -type int -range_min 0
```

```
pt_shell> define_user_attribute attr_ir3 \
            -classes cell -type int -range_max 100
```

In the following example, define attribute attr_oo for cells. The attribute is a string, but can only be set to A, B, C, or D.

```
pt_shell> define_user_attribute attr_oo \
            -classes cell -type string -one_of {A B C D}
```

After these attributes have been defined, you can list the attribute definitions using the **list_attribute** command. Notice how the 'attr_i' attribute is marked as importable from db, meaning a design or library database.

```
pt_shell> list_attributes

****************************************
Report : List of Attribute Definitions
 ...
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from db (for user-defined)

Attribute Name   Object  Type      Properties  Constraints
-------------------------------------------------------------
attr_i           cell    int       U,I
attr_ir1         cell    int       U           0 to 100
attr_ir2         cell    int       U           >= 0
attr_ir3         cell    int       U           <= 100
attr_oo          cell    string    U           A, B, C, D
attr_s           cell    string    U
attr_s           net     string    U
design1          design  int       U,I
design1          cell    int       U
```

SEE ALSO

```
get_attribute(2)
list_attributes(2)
read_db(2)
read_ddc(2)
read_milkyway(2)
remove_user_attribute(2)
report_attribute(2)
set_user_attribute(2)
```

derive_clocks

Creates clocks on source pins in design.

SYNTAX

```
string derive_clocks
   -period period_value
   [-waveform edge_list]
```

Data Types

```
period_value      float
edge_list         list
```

ARGUMENTS

-period *period_value*

Specifies the clock period of the automatically derived clocks. The clock period has a value greater than or equal to zero (value >= 0).

-waveform *edge_list*

Specifies the rise and fall edge times of the clock, in library time units, over an entire clock period. It defines the clock edge specification. The first time that is listed is a rising transition; typically the first rising transition after time zero. There must be an even number of increasing times and alternating rise and fall times. If you do not specify an *edge_list* value, the command assumes a default waveform that has a rise edge of *0.0* and a fall edge of *period_value*/2.

DESCRIPTION

Creates clocks on source pins in design. This command finds the clock source ports and pins that must have clocks defined so that every register clock pin has a clock. Then on each source set, the command creates a clock in the same way as the **create_clock** command does. The clocks are created with the specified waveform or period. For a description of commands that use clock objects, see the **create_clock** command man page.

EXAMPLES

The following example creates a clock on the source pin with a clock period of 20:

```
pt_shell> derive_clocks -period 20
```

SEE ALSO

```
create_clock(2)
get_clocks(2)
group_path(2)
remove_clock(2)
report_clock(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_input_delay(2)
set_output_delay(2)
```

disconnect_net

Disconnects a net from specified pins or ports, or from all pins and ports.

SYNTAX

```
status disconnect_net
  -all
  net
  object_spec
```

Data Types

```
net          string
object_spec  list
```

ARGUMENTS

-all

Indicates that all pins and ports are to be disconnected from *net*. The **-all** and *object_spec* options are mutually exclusive.

net

Specifies the name of the net to be disconnected.

object_spec

Specifies a list of pins or ports to be disconnected from *net*. The **-all** and *object_spec* options are mutually exclusive.

DESCRIPTION

The **disconnect_net** command disconnects pins and ports from a *net* in the current design. Like all other netlist editing commands, for the **disconnect_net** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. the **disconnect_net** command returns a 1 if successful and a 0 if unsuccessful.

The *net* option and each pin and port specified in the *object_spec* option must be in scope; that is, at or below the current instance. There are three rules for **disconnect_net**. They are:

- You cannot disconnect an unconnected pin or a port.

- You cannot disconnect a pin or a port that is not connected to *net*.
- You cannot disconnect a hierarchical pin or a port that has the same name as *net*.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is re-linked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following example, the first command attempts to connect port in1 to net new_net1, but in1 is already connected to a net. The second command determines the net to which in1 is connected; the third command disconnects the port from its net. The fourth and final command connects in1 to new_net1.

```
pt_shell> connect_net new_net1 [get_ports in1]
Error: Could not connect 'in1' to 'new_net1'
        Object is already connected - disconnect it first (NED-042)
```

```
Error: No changes made. (NED-040)
0
pt_shell> set pnet [get_nets -of_objects [get_ports in1]]
{"in1"}
pt_shell> disconnect_net $pnet [get_ports in1]
Information: Disconnected 'in1' from 'net1'. (NED-019)
1
pt_shell> connect_net new_net1 [get_ports in1]
Information: Connected 'in1' to 'new_net1'. (NED-018)
1
```

SEE ALSO

connect_net(2)
create_cell(2)
create_net(2)
size_cell(2)
swap_cell(2)
write_changes(2)

drive_of

Determines the drive resistance of the specified library cell pin. The **drive_of** command is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
float drive_of
   [-rise]
   [-fall]
   [-wire_drive]
   [-piece val]
   lib_cell_pin
```

Data Types

val	string
lib_cell_pin	string

ARGUMENTS

-rise

Gets rise drive value.

-fall

Gets fall drive value.

-wire_drive

Not supported in PrimeTime.

-piece *val*

Not supported in PrimeTime.

lib_cell_pin

Specifies the name of the library cell pin whose drive value is to be returned. This value can be either the drive resistance or a collection that contains the library cell pin.

DESCRIPTION

The **drive_of** command returns the drive resistance of the given library cell pin. If neither the *-rise* nor *-fall* option

is specified, the greater value of rise and fall is returned.

The **drive_of** command exists in PrimeTime for compatibility with Design Compiler. Complete information about the **drive_of** command can be found in the Design Compiler documentation. The supported method for getting the capacitance of a library cell pin is by using the **get_attribute** command with either the **drive_resistance_rise** or **drive_resistance_fall** attribute.

SEE ALSO

`get_attribute(2)`

echo

Echos arguments to standard output.

SYNTAX

```
string echo
   [-n]
   [arguments]
```

Data Types

```
arguments      string
```

ARGUMENTS

-n

Suppresses the new line. By default, **echo** adds a new line.

arguments

Specifies the arguments to be printed.

DESCRIPTION

The **echo** command prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a new line, but if the **-n** option is specified, multiple **echo** command outputs are printed onto the same output line.

The **echo** command is used to print out the value of variables and expressions in addition to text strings. The output from **echo** can be redirected using the **>** and **>>** operators.

EXAMPLES

The following are examples of using the **echo** command:

```
prompt> echo
"Running version" $sh_product_version
```

```
Running version v3.0a

prompt> echo -n "Printing to" [expr (3 - 2)] > foo
prompt> echo " line." >> foo
prompt> sh cat foo
Printing to 1 line.
```

SEE ALSO

sh(2)

enable_primetime_icc_consistency_settings

Sets all recommended variables with values for consistency with IC Compiler.

SYNTAX

```
string enable_primetime_icc_consistency_settings
   [-all]
```

ARGUMENTS

-all

Applies the recommended values to all variables, including flow-specific variables. If you do not use this option, flow-specific variables are not adjusted or reported.

DESCRIPTION

The **enable_primetime_icc_consistency_settings** command sets all common variables to have values that are recommended for consistency. By default, the flow-specific and feature gap variables are ignored.

EXAMPLES

```
pt_shell> enable_primetime_icc_consistency_settings
set rc_degrade_min_slew_when_rd_less_than_rnet false
# rc_degrade_min_slew_when_rd_less_than_rnet changed to true
set timing_clock_gating_propagate_enable true
# timing_clock_gating_propagate_enable changed to false
```

error_info

Prints extended information on errors from the last command.

SYNTAX

```
string error_info
```

ARGUMENTS

This **error_info** command has no arguments.

DESCRIPTION

The **error_info** command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the **error_info** command can help you to focus on the exact line in a block that caused the error.

EXAMPLES

This example shows how **error_info** can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '$s == "a" '.

```
prompt> foreach s $my_list {
?         if { s == "a" } {
?           echo "Found 'a'!"
?         }
?       }
Error: syntax error in expression " s == "a" "
       Use error_info for more info. (CMD-013)
shell> error_info
Extended error info:
syntax error in expression " s == "a" "
    while executing
"if { s == a } {
     echo "Found 'a'"
}"
    ("foreach" body line 2)
    invoked from within
"foreach s [list a b c] {
   if { s == a } {
```

```
        echo "Found 'a'"
    }
    }"
     -- End Extended Error Info
```

estimate_clock_network_power

Virtually generate a clock tree and estimate its power.

SYNTAX

```
string estimate_clock_network_power
   lib_cell
   [-max_fanout fanout]
   [-wire_load_model wire_load_name]
   [-library lib_name]
   [-input_transition transition]
   [-clocks clock_list]
   [-include_registers]
   [-ignore_clock_gating]
```

Data Types

lib_cell	string
fanout	int
wire_load_name	string
lib_name	string
transition	float
clock_list	list

ARGUMENTS

lib_cell

Specifies the buffer or inverter to be used to build the clock tree.

-max_fanout *fanout*

Specifies the maximum fanout number that the specified buffer can drive. The default number is 32.

-wire_load_model *wire_load_name*

Specifies the wire load model to be used to compute the wire capacitance for the nets in the clock tree. The default is the wire load model set to the design.

-library *lib_name*

Specifies the library where the specified wire load model is defined.

-input_transition *transition*

Specifies the clock input transition. If not specified, use the clock transition annotated to the clock by the **set_clock_transition** command. If there is no annotation either, the default to 0.

-clocks *clock_list*

Specifies the clocks to be estimated for clock tree power. The default is all the clocks in the design.

-include_registers

Indicates the power of registers driven by the clock is also included in the power report.

-ignore_clock_gating

Indicates that existing clock tree cells like the clock gating cells are ignored when building the clock tree.

DESCRIPTION

The **estimate_clock_network_power** command virtually creates a clock tree for each clock and calculates power for the generated clock tree. The clock trees are built on the fly and the original design is not touched or modified. The following describes how the clock tree is built:

Find all the nets in the clock network. For each net in the clock network, insert buffers to build a clock tree, so that the fanout of each buffer in the clock tree is no bigger than the max fanout. The command tries to build a tree as balanced as possible such that the delays from the root to the leafs do not vary a lot. All the driven registers are put at the same and lowest level of the tree. The deviation of buffer fanouts at the same tree level is kept as small as possible.

The output load of each buffer is calculated using wire load model. The input transition of each buffer propagates from the root of the tree. The switching activity for each inserted buffer is equal to clock frequency or from annotation (SAIF or set_switching_activity).

If there are clock gating cells in the original clock network, the command builds a separate tree based on the fanout of the clock gating cell using the same rule for the clock tree, and this separate tree is considered as a branch of the whole clock tree, and also balances with other branches. Then the clock gating effect is accounted for by the annotated or, if not annotated, propagated switching activity. In order for you to investigate savings from clock gating, the command has the *-ignore_clock_gating* option to estimate the clock tree power by ignoring the clock gating cells and compare with the power that considers the clock gating effect. The power consumed by the registers driven by the clock is also included in the report if the *-include_registers* option is specified.

EXAMPLES

In the following example, clock tree power is estimated for clock CLK by building the clock tree using the buffer buf1a1 of library ssc_core_typ and wire load model in the design is used to calculate the wire capacitance. The clock input transition is 0.2.

```
pt_shell> create_clock -name CLK -period 10 clk
pt_shell> set_clock_transition 0.2 CLK
pt_shell> estimate_clock_network_power ssc_core_typ/buf1a1
```

SEE ALSO

```
report_power(2)
```

exit

Terminates the application.

SYNTAX

```
integer exit
   [exit_code]
```

Data Types

exit_code	integer

ARGUMENTS

exit_code

Specifies the return code to the operating system. The default value is 0.

DESCRIPTION

This command exits from the application. You have the option to specify a code to return to the operating system.

EXAMPLES

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify (**echo**) the return code as shown.

```
prompt> exit 5

% echo $status
5
```

SEE ALSO

```
quit(2)
```

export_power_management_mtcmos_locations

Exports the power management location (PML) file for the specified cell.

SYNTAX

```
status export_power_management_mtcmos_locations
   -cell cell_name
   -library library_name
   -output_file pml_file_name
```

Data Types

```
string cel_name
string library_name
string pml_file_name
```

ARGUMENTS

-cell *cel_name*

Specifies the name of the Milkyway CEL design whose power management cell location information is to be exported.

-library *library_name*

Specifies the Milkyway design library in which the Milkyway CEL view exists.

-output_file *pml_file_name*

Specifies the file name of the PML file to be exported.

DESCRIPTION

This command exports the PML file for the specified cell. The PML file describes the on-resistance source and drain locations of power management cells.

EXAMPLES

The following example exports the PML file for the PMCell cell, and saves the output PML file in the name pml.

```
pr_shell> export_power_management_mtcmos_locations -library MyLib \
             -cell PMCell -output_file pml.txt
```

SEE ALSO

```
import_power_management_mtcmos_locations(2)
remove_power_management_mtcmos_locations(2)
```

extract_model

Generates a timing model or a timing and power model from the gate-level netlist of a design.

SYNTAX

```
status extract_model
   [-context_borrow]
   [-latch_level levels]
   -output file_name
   [-format format_list]
   [-script_format ptsh | dcsh]
   [-library_cell]
   [-ignore_boundary_parasitics]
   [-test_design]
   [-validate valid_list]
   [-arc_types arc_list]
   [-noise]
   [-power]
```

Data Types

```
levels            int
file_name         string
format_list       list
valid_list  list
scenario_name     string
arc_list          list
```

ARGUMENTS

-context_borrow

Determines which latches on an interface borrow, based on all input port arrival times, clock definitions, and clock latencies specified at the time of model extraction. The actual time borrowed by a latch is not used by model extraction; only whether a latch borrows or not. During model extraction, the tool traces through a borrowing latch and stops at a nonborrowing latch. Use this option if your design contains latches on its interface. The **-context_borrow** and **-latch_level** options are mutually exclusive. If neither is specified, **-context_borrow** is assumed.

-latch_level *levels*

Specifies the maximum number of levels of latches allowed at the interface of a design. Whether a latch borrows is still determined by the same approach as the **-context_borrow** option. This option specifies an upper limit; it does not cause a latch to borrow through even if its level from the port is smaller than the limit. Use this option only after you have verified that all latches at the interface of a design borrow, if they are at a certain level, and you want to limit the numbers of borrow to consider for model extraction. The **-context_borrow** and **-latch_level** options are mutually exclusive. If neither is specified, **-context_borrow**

is assumed.

-output *file_name*

Specifies the root name to use in deriving the names of the generated output files; the default is the current design name. The root name is appended with an appropriate extension (.db, .data, .mod, or .lib), depending on the value of the **-format** option.

-format *format_list*

Specifies a list of one or more output formats for the generated model. If the list contains more than one item, they must be enclosed in braces. Allowed values are as follows:

- **db** (the default) - Generates the output file in the Synopsys database format with the _lib.db extension. The output contains the database library description of the core cell of the model.
- **lib** - Generates the output file in the Synopsys Library Compiler format with the .lib extension.

-script_format ptsh | dcsh

Specifies the output format for the script. Allowed values are **ptsh** for pt_shell, **dcsh** (the default) for dc_shell.

-library_cell

Indicates that the model is to be generated as a Liberty library cell, named *current_design name*. The option is also the default behavior, so is not needed.

-ignore_boundary_parasitics

Indicates that detailed parasitic annotations present on the boundary nets of the design are not to be included in the model. The **-ignore_boundary_parasitics** option causes the effects of the boundary net parasitics to be excluded from the library cell model.

-test_design

Indicates that a test design DB instantiating the model library cell is to be generated. The design name is *current_design*_test and the DB file is *current_design*_test.db. If you use the **-test_design** option, you must also use the **-library_cell** option.

-validate *valid_list*

Specifies what should be automatically validated after the model is created. The allowed values are **timing** and **noise**. If you use the **-validate** option, and the **hier_modeling_version** variable is set to 2.0, the model is automatically validated after the model is created.

-arc_types *arc_list*

Specifies a list of one or more arc types to be included when extracting a model; only the specified types are extracted. By default, all arc types are included. Allowed values are **min_seq_delay**, **max_seq_delay**, **min_combo_delay**, **max_combo_delay**, **setup**, **hold**, **recovery**, **removal**, and **pulse_width**. If the list contains more than one item, enclose the items in braces.

-noise

Specifies that noise features should be added to the model. If the option is specified, and no noise information is available in the cell's libraries, and you have specified none, only steady-state resistance is written by using an estimation. This option performs an implicit **update_noise**, if necessary.

-power

Specifies that power features should be added to the model and performs an implicit **update_power**, if necessary. The **power_enable_analysis** variable must be set to **true**, and you must have a PrimeTime PX

license.

DESCRIPTION

The **extract_model** command generates a static timing model for the current design from its gate-level netlist. The generated model exhibits the same timing characteristics as the original design and therefore is used instead of the gate-level netlist for purposes of static timing analysis. To generate noise and power information, use the **-noise** and **-power** options, respectively.

By default, the **extract_model** command generates a context-independent model, in which the timing changes as the actual context varies at the top level. The timing model tracks the original block interface timing. However, the extracted model can become context-dependent. Especially when borrowing latches exist at the block interface, the generated model is valid only for the clock waveforms, latencies, input delays, and output delays specified for the extraction. When the context changes sufficiently to cause different borrowing on the interface, the timing model does not represent the new borrowing. This default behavior is specified by the **-context_borrow** option.

To override the default behavior, use the **-latch_level** option. When you use this option, changes in clock waveforms, output capacitive loads, input drives, input delays, and output delays result in different timing exhibited by the model.

The generated model is captured as a Liberty syntax library cell.

EXAMPLES

The following example generates a library cell model for the current design. The model is generated for the operating condition set on the design. In addition, the time borrowed by each level-sensitive latch is locked at its current value in the generated model. Two files are generated: DMA_model_lib.db and DMA_model_constr.pt

```
pt_shell> extract_model -context_borrow -output DMA_model
```

The following example generates a model in Synopsys DB format for the current design. The generated model is a library cell contained in model2_lib.db. The file model2_test.db is also generated; this contains the test design that instantiates the model core cell, but it is not a part of the model. model2_test.db can be used to obtain timing and model reports in PrimeTime.

```
pt_shell> extract_model -library_cell \
            -test_design -output model2 -format {db}
```

SEE ALSO

```
extract_model_capacitance_limit(3)
extract_model_clock_latency_arcs_include_all_registers(3)
extract_model_clock_transition_limit(3)
extract_model_data_transition_limit(3)
extract_model_db_naming_compatibility(3)
extract_model_enable_report_delay_calculation(3)
extract_model_gating_as_nochange(3)
extract_model_include_clock_tree_pulse_width(3)
```

```
extract_model_include_ideal_clock_network_latency(3)
extract_model_include_upf_data(3)
extract_model_keep_inferred_nochange_arcs(3)
extract_model_lib_format_with_check_pins(3)
extract_model_merge_clock_gating(3)
extract_model_noise_iv_index_lower_factor(3)
extract_model_noise_iv_index_upper_factor(3)
extract_model_noise_width_points(3)
extract_model_num_capacitance_points(3)
extract_model_num_clock_transition_points(3)
extract_model_num_data_transition_points(3)
extract_model_num_noise_iv_points(3)
extract_model_num_noise_width_points(3)
extract_model_single_pin_cap(3)
extract_model_single_pin_cap_max(3)
extract_model_split_partial_clock_gating_arcs(3)
extract_model_status_level(3)
extract_model_suppress_three_state(3)
extract_model_upf_supply_precedence(3)
extract_model_use_conservative_current_slew(3)
extract_model_with_3d_arcs(3)
extract_model_with_ccs_timing(3)
extract_model_with_clock_latency_arcs(3)
extract_model_with_min_max_delay_constraint(3)
extract_model_write_case_values_to_constraint_file(3)
hier_modeling_version(3)
timing_clock_gating_propagate_enable(3)
timing_disable_clock_gating_checks(3)
timing_enable_preset_clear_arcs(3)
```

extract_supply_parasitics

Extracts supply net parasitics for the specified supply nets.

SYNTAX

```
extract_supply_parasitics
   supply_nets
   [-noexpand ]
   [-include_capacitances integer]
   [ -wait | -nowait ]
```

Data Types

supply net collection *supply_nets*

ARGUMENTS

supply_nets

Lists the supply nets to be extracted. Each element of the list can be a supply net collection or a name pattern. The list of supply nets is automatically expanded to include the supply nets that are connected by MTCMOS switch cells to those listed (that is, power management or PM cells). This automatic expansion is suppressed by the *-noexpand* option.

-noexpand

Prevents the listed *supply_nets* from being expanded to include the supply nets that are connected to the listed supply nets by MTCMOS switch cells (also known as power management or PM cells). By default, the list of supply nets is expanded.

-include_capacitances

When it is specified with value 1, the command will perform RC extraction on the PG nets. If it is specified with value 0, the command will perform R only extraction, where capacitance extraction is not included. By default, the command performs RC extraction.

-wait | -nowait

The command waits for the completion of the extraction if the *-wait* option is specified. The command returns without waiting for the completion of the extraction if the *-nowait* option is specified. By default, PrimeRail determines which option to use based on the estimated memory usage of the extraction, timing analysis and power analysis, and the available virtual memory of the machine.

DESCRIPTION

This command performs parasitic extraction for the specified power and ground supply nets. The design must be loaded into the session using the **open_mw_cel** command before **extract_supply_parasitics** is executed.

The list of supply nets on which to perform parasitic extraction is automatically expanded to include the supply nets that are related to the specified nets by being connected through MTCMOS switch cells (also known as PM cells). This automatic supply net expansion behavior saves you from having to identify the full set of related supply nets yourself. You can suppress this auto expansion by using the *-noexpand* option.

Configure the necessary extraction options with the PrimeRail **set_extract_supply_net_options** command. Refer to the **set_extract_supply_net_options** man page for details.

The supply net extraction data is saved in the RAIL_DATABASE/pgext/para directory.The extraction data should be read into memory by the PrimeRail **read_supply_parasitics** command. See the **read_supply_parasitics** man page for details.

EXAMPLES

The following example shows how to extract the supply nets VDD and VSS. The command returns without waiting for the completion of the extraction.

```
pr_shell> extract_supply_parasitics -nowait {VDD VSS}
```

The following example shows how to extract the supply nets VDD and VSS. The command waits for the completion of the extraction.

```
pr_shell> extract_supply_parasitics -wait {VDD VSS}
```

SEE ALSO

```
open_mw_cel(2)
set_extract_supply_net_options(2)
report_extract_supply_net_options(2)
```

filter_collection

Filters an existing collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection filter_collection
   [-regexp]
   [-nocase]
   collection1
   expression
```

Data Types

collection1	collection
expression	string

ARGUMENTS

-regexp

Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

-nocase

Makes the pattern match case-insensitive.

collection1

Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as **false** by the conditional *expression* value. Substitute the collection you want for *collection1*.

expression

Specifies an expression with which to filter *collection1*. Substitute the string you want for *expression*.

DESCRIPTION

Filters an existing collection, resulting in a new collection. The base collection remains unchanged. In many cases, application commands that create collections support a *-filter* option that filters as part of the collection process, rather than after the collection has been made.

This type of filtering is almost always more efficient than using the **filter_collection** command after a collection has been formed. The **filter_collection** command is most useful if you plan to filter the same large collection many times using different criteria.

The **filter_collection** command results in either a new collection or an empty string. A resulting new collection contains the subset of the objects in the input *collection1*. A resulting empty string (the empty collection) indicates that the *expression* filtered out all elements of the input *collection1*.

The basic form of the conditional expression is a series of relations joined together with AND and OR operators. Parentheses () are also supported. The basic relation contrasts an attribute name with a value through a relational operator. For example,

```
is_hierarchical == true and area <= 6
```

If an attribute is a collection attribute that contains a single object, then you can query an attribute value from that object using "." to name the attribute on the other object. The "." operator can be chained. For example.

```
owner.is_hierarchical == true and owner.area <= 6
shape.net.name==reset
```

The value side of a relation can be a simple string, quoted string, or an attribute name prefixed with "@". For example:

```
input_delay<=@output_delay
input_delay<=0.24
name=="@literal_name"
```

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only either *true* or *false*.

Existence relations determine if an attribute is defined or not defined for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
!defined
```

These operators apply to any attribute as long as it is valid for the object class.

Collection attributes support a special sizeof(attrName) operator that returns the number of objects in the attribute. If the attribute is not set then 0 is returned.

The **filter_collection** command has a *-regexp* option that uses regular expressions when matching for string attributes. Regular expression matching is done in the same way as in the Tcl **regexp** command. When using the -

regexp option, take care in the way you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the *-nocase* option.

EXAMPLES

The following example from PrimeTime creates a collection of only hierarchical cells.

```
pt_shell> set a [filter_collection [get_cells *] \
"is_hierarchical == true"]
{"Adder1", "Adder2"}
```

The following example from PrimeTime uses existence operators to create a collection of all nonmoded cell timing_arc objects in the current design.

```
pt_shell> set b [filter_collection \
            [get_timing_arcs -of_objects [get_cells *]] \
            "undefined(mode)"]
```

```
The following example finds all cells from {U0(1) U1(1) U0(0) U1(0)}
with the index (0). Notice the round brackets used for the index.

pt_shell> filter_collection -regexp [get_cells] {full_name =~ "U.*\(0\)"}
{U0(0) U1(0)}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
pt_shell> filter_collection [get_ports] -regexp {name=~ "in\[(1[8-9]|2[0-3])\]"}
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the U2 and U4 cells, using the -nocase option:

```
pt_shell> filter [get_cells] -regexp -nocase {name =~ u[24]}
{U2 U4}
```

SEE ALSO

```
collections(2)
regexp(2)
```

find

The **find** command, used to create a collection of design objects, is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
string find
   type
   [-hierarchy]
   [-flat]
   [object_list]
```

Data Types

type	string
object_list	list

ARGUMENTS

type

Specifies the object class to find. The type value can be *design*, *port*, *net*, *cell*, *pin*, *clock*, *library*, *lib_cell*, or *lib_pin*.

-hierarchy

Find objects throughout the hierarchy.

-flat

Not supported by PrimeTime.

object_list

Specifies the patterns to match.

DESCRIPTION

The **find** command creates a collection of objects. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the **find** command can be found in the Design Compiler documentation. The supported method for creating collections of objects is to use **get*** commands (for example, the **get_cells** command or the **get_pins** command).

EXAMPLE

The following example finds the N1 net in the current design:

```
pt_shell> find net N1
{N1}
```

The following example uses the * (asterisk) wildcard character to find all cells in the current design that begin with U:

```
pt_shell> find cell U*
{U0, U1, U2, U3, U4, U5, U6, U7}
```

For more examples, see the **find** command in Design Compiler.

SEE ALSO

```
get_cells(2)
get_clocks(2)
get_designs(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_pins(2)
get_ports(2)
```

find_objects

Finds logical hierarchy objects within a scope. This is a UPF query command.

SYNTAX

```
[-collection find_objects]
[-pattern pattern_string]
[-object_type inst | port | net | model]
[-direction in | out | inout]
[-transitive true | false]
[-non_leaf]
[-leaf_only]
[-exact]
scope
```

Data Types

```
pattern_string      string
scope               string
```

ARGUMENTS

-pattern *pattern_string*

Specifies a search pattern. By default, the pattern is treated as an Tcl glob expression.

-object_type inst | port | net | model

Searches for only the specified object type:

- **inst** - Cell instance
- **port** - Design port
- **net** - Net
- **model** - Model or library cell

-direction in | out | inout

Specifies the direction of the searched ports if the **-object_type port** option is used.

-transitive true | false

Controls whether the descendants of the elements are included in the search:

- **false** (default) - Does not search descendants.
- **true** - Searches descendants.

-non_leaf

Searches only nonleaf design elements (elements that have child elements).

-leaf_only

Searches only leaf-level design elements (elements without child elements). By default, both leaf and nonleaf design elements are searched.

-exact

Searches for an exact match of the *pattern_string* argument; disallows wildcard expansion on the *pattern_string*.

scope

Specifies the scope in which to search for logical hierarchy objects.

DESCRIPTION

The **find_objects** command finds logical hierarchy objects within a scope.

Multicorner-Multimode Support

This command has no dependency on scenario specific information.

EXAMPLES

The following examples show the usage of the command with different options.

```
prompt> find_objects a -pattern * -object_type inst \
         -transitive TRUE -non_leaf
a/b1 a/b2

prompt> find_objects a/b2 -pattern b_? -object_type net
a/b2/b_i a/b2/b_o

prompt> find_objects a/b1 -pattern * -object_type port \
        -transitive TRUE -direction out
a/b1/c1/Z a/b1/c2/Z a/b1/b_o

prompt> find_objects . -object_type model -pattern b* -non_leaf \
        -transitive TRUE
a/b1 a/b2 b1

prompt> find_objects . -object_type model -pattern b* -non_leaf \
        -transitive TRUE -exact
Warning: Can't find module or lib cell 'b*' in design 'top'. (UID-95)

prompt> find_objects . -object_type model -pattern bot -non_leaf \
        -transitive TRUE -exact
a/b1 a/b2 b1

For black box bus port
prompt> find_objects . -pattern A* -object_type port
A[0] A[1] A[2]
```

SEE ALSO

```
get_cells(2)
get_nets(2)
get_ports(2)
set_scope(2)
```

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

```
string foreach_in_collection
   itr_var
   collections
   body
```

Data Types

```
itr_var          string
collections      list
body             string
```

ARGUMENTS

itr_var

Specifies the name of the iterator variable.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute per iteration.

DESCRIPTION

The **foreach_in_collection** command is used to iterate over each element in a collection. You cannot use the Tcl-supplied **foreach** command to iterate over collections because the **foreach** command requires a list, and a collection is not a list. Also, using the **foreach** command on a collection causes the collection to be deleted.

The arguments for the **foreach_in_collection** command parallel those of the **foreach** command: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. All arguments are required.

Note: The **foreach_in_collection** command does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an argument also accepts *itr_var* because they are of the same data type (collection).

You can nest the **foreach_in_collection** command within other control structures, including another **foreach_in_collection** command.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration will be deleted. The **foreach_in_collection** command is safe for such operations. If a command in the body causes the collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, then apply one or more commands to that collection. If the order of operations does not matter, the following are equivalent. The first is an example without iterators.

```
set s [get_cells {U1/*}]
command1 $s
command2 $s
unset s
```

The following is a similar approach using the **foreach_in_collection** command:

```
foreach_in_collection itr [get_cells {U1/*}] {
  command1 $itr
  command2 $itr
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, though both produce the same results if the commands are order-independent.

EXAMPLES

The following example from PrimeTime removes the wire load model from all hierarchical cells in the current instance.

```
pt_shell> foreach_in_collection itr [get_cells *] {
?             if {[get_attribute $itr is_hierarchical] == "true"} {
?               remove_wire_load_model $itr
?             }
?          }
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

SEE ALSO

collections(2)
foreach(2)

generate_rail_macro_model_probes

Generates a probe file that records how a node statue changes for circuit simulation.

SYNTAX

```
status generate_rail_macro_model_probes
   -spice_netlist file_name
   -power_nets power_net_names
   -ground_nets ground_net_names
   [-switch_supply_net_pairs net_pairs]
   [-prefix prefix_for_probing_nodes]
   [-output file_name]
```

Data Types

```
string file_name
string prefix_for_probing_nodes
list power_net_names
list ground_net_names
list net_pairs
```

ARGUMENTS

-spice_netlist *file_name*

Specifies the SPICE netlist that is generated by an LVS tool, such as the IC Validator tool.

-power_nets *power_net_names*

Specifies the power nets whose node status is to be recorded.

-ground_nets *ground_net_names*

Specifies the ground nets whose node status is to be recorded.

-switch_supply_net_pairs *net_pairs*

Specifies the internal and external net pairs for fine-grain memory cells being analyzed, in the following syntax:

-switch_supply_net_pairs {{internal_net external_net}}

-prefix *prefix_for_probing_nodes*

Specifies the prefix for probing nodes.

-output *file_name*

Specifies the name of probe node file to be generated. The default name is rmm.probe.

DESCRIPTION

This command generates probing nodes for running circuit simulation.

EXAMPLES

The following example generates a probe file that records node changes for the PG nets, VDD and VSS, based on the specified SPICE netlist, test.sp.

pr_shell> **generate_rail_macro_model_probes -spice_netlist test.sp -power_nets { VDD } -ground_nets { VSS } -output mm.probes**

SEE ALSO

`create_rail_macro_model(2)`

generate_reduced_core_model

Generates reduced core model for current design, specified PG nets.

SYNTAX

```
status generate_reduced_core_model
   [-static | -dynamic]
   [-start_time time]
   [-end_time time]
   [-step_size time]
   [-spice_dump_file spice_file_name]
   [-probing_points_file probing_points_file_name]
   [-merge_ports_by_net | -merge_port_file
   probing_points_file_name]
   supply_nets
```

Data Types

```
float time
string file_name
float value
list supply_nets
```

ARGUMENTS

-static Specifies the reduced core model to be *static*. If niether "-static" nor "-dynamic" is specified, this command takes types from **update_current** command. This option is incompatible with the *-dynamic* option.

-dynamic Specifies the reduced core model to be *dynamic*. If neither "-static" nor "-dynamic" is specified, this command takes type from **update_current** command. This option is incompatible with the *-static* option.

-start_time *time* Specifies the start time for the dynamic analysis, time unit is second. The start time is the time relative to the reference time (or the vcd start time) reported by command **update_currents**.

-end_time *time* Specifies the end time for the dynamic analysis, time unit is second. The end time is the time relative to the reference time (or the vcd start time) reported by command **update_currents**.

-step_size *time* Specifies the simulation time step size to use during dynamic analysis, time unit is second.

-spice_dump_file *spice_file_name* Specifies the spice file name that reports the generated reduced core model. If no spice dump file specified, the command will dump the core model to spice file "CoreModel.sp".

-probing_points_file *probing_points_file_name* Specifies the input file name that stores probing points. This argument is optional. It is only needed when the user wants to do internal node probing. Each line in this file represent one probing points, specified by layer, x, y. The coordinate x, y are float numbers in unit of um.

-merge_ports_by_net Specifies the user needs to merge ports into groups by their net. With this option specified, all

the ports on same net will be merged into one port in the reduced core model. This option should not be specified at the same time as the option -merge_port_file *filename*.

-merge_port_file *merge_ports_file_name* Specifies the user needs to merge ports into groups according to the grouping information specified in *merge_ports_file_name*. The format of merge port file is same as tap file, with one additional column specifying the groupId. For example, for each tap point that need to be grouped, the line for this tap point would be *netname layer xcoord ycoord groupId*. The values for *xcoord* and *ycoord* are in unit of um, same as in tap file. The *groupId* value has to be non-negative. The tap points to be merged into same group will have same groupId. We don't need to specify every tap point's group in this file. For tap points that are not specified in this merge port file, we will not group them with other tap points. This option should not be specified at the same time as the option -merge_ports_by_net.

supply_nets Lists the supply nets on which voltage drop analysis is performed. Each element of the list can be a supply net collection or a name pattern.

DESCRIPTION

The command generates the reduced core model for given PG nets of the current design, using the current tap points that are specified with the **create_taps** command as boundary ports. The generated core model consists two parts: the port currents represented as values or waveforms in time domain, and reduced PG network represented in frequency domain as S-parameter model. The calculation of the port currents is based on static (that is, average) power and ground pin currents and static rail analysis if the *-static* option is specified, or is based on dynamic (time-varying) PG pin currents and dynamic rail analysis if the *-dynamic* option is specified.

Note that supply net parasitics must be read into the session by command **read_supply_parasitics** before this command is executed. Also, PG pin currents must be available before executing this command by either **update_currents** or **set_rail_current**. And the tap points should have been created before lauching this command, otherwise there is no boundary condition.

EXAMPLES

The following example creates tap points from imported tap file, reads in the supply net parasitics, specifies PG pin currents by update_currents, then generates static reduced core model:

```
pr_shell> create_taps -import tapfile
pr_shell> read_supply_parasitics VDD
pr_shell> update_currents -static
pr_shell> generate_reduced_core_model VDD -static -spice_dump_file rcmstatic.sp
pr_shell> ************************************
The following example specifies PG pin currents by set_rail_current, then generates dynamic reduced c

pr_shell> set_rail_current -waveform {0.0 0.0 0.1 -500.0 0.2 -1000.0 0.5 -1000.0 0.6 -500.0 0.7 0.0 5

pr_shell> generate_reduced_core_model VDD -dynamic -start_time 0 -end_time 50e-9 -step_size 1e-10 -sp
```

SEE ALSO

```
create_taps(2)
```

```
get_taps(2)
read_supply_parasitics(2)
update_currents(2)
set_rail_current(2)
report_taps(2)
```

get_alternative_lib_cells

Creates a collection of library cells that can be used to replace or "size" a specified cell in the current design.

SYNTAX

```
collection get_alternative_lib_cells
  [-current_library]
  [-libraries lib_spec]
  [-base_names]
  cell
```

Data Types

```
lib_spec      list
cell          string
```

ARGUMENTS

-current_library

If this option is specified, then PrimeTime searches for library cells in the cell's current library or library cell's current library only. You cannot specify this option with the **-libraries** option.

-libraries *lib_spec*

Specifies a list of libraries from which to select alternative library cells. The default is to select from all libraries in the link path. The *lib_spec* option can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the **get_libs** command. You cannot specify this option with the **-current_library** option.

-base_names

Specifies that a list of library cell base names is to be returned. Multiple library cells can have the same base name, however the list is stripped of these duplicate names. This option is enabled by default if you invoke PrimeTime with the **-multi_scenario** option.

cell

Specifies a cell or a library cell for which alternative library cells is required.

DESCRIPTION

The **get_alternative_lib_cells** command creates a collection of library cells that can be used to replace or

"size" a specified cell or library cell. A library cell is included in the collection only if it satisfies the same criteria used by the **size_cell** command to determine whether a library cell can be used to replace a cell in the current design.

If the **-current_library** option is specified, then PrimeTime searches for library cells in the cell's current library or library cell's current library only. If the **-libraries** option is specified, then PrimeTime searches for library cells in the libraries contained within the *lib_spec* option only.

Alternatively, if neither of the above options are specified, PrimeTime searches for the library cell in the following sources in this order:

1. The cell's current library.
2. The **link_path_per_instance** variable.
3. The **link_path** variable, if no **link_path_per_instance** specification pertains to the cell.

To see a comparison of the slack and design cost within a set of alternative library cells, use the **report_alternative_lib_cells** command.

Equivalent Library Cells

Two library cells are compatible for sizing if all of the following restrictions are satisfied. The exact same set of restrictions applies to the **size_cell**, **get_alternative_lib_cells** and **report_alternative_lib_cells** commands.

- The library cells must be functionally equivalent. At least one of the following must be true, and none can be false:
  - If defined, the **function_id** attribute must match. This attribute is created by Library Compiler and is available on **lib_cell** objects.
  - If defined, the **mog_func_id** attribute must match. This attribute is created by Library Compiler for multiple output gate (MOG) cells and is available on **lib_cell** objects.
  - If defined, the **user_function_class** attribute must match. This is a user-defined functional equivalency attribute for cells that are too complex for Library Compiler to compute functional information for. This attribute is available on **lib_cell** objects.
- The number of pins, ordering of pins, pin directions, and internal pin names must match.
- The number of enabled library cell arcs, arc types and sense must match, including any conditional or "when" expressions.

EXAMPLES

The following example shows how to find alternative library cells for a cell in the design. Here, *u_max* is an instance of class_max/OR3 and *u_min* is an instance of class_min/OR3.

```
pt_shell> set link_path [list * class_max.db]
* class_max.db
pt_shell> set_min_library class_max.db -min_version class_min.db
Loading db file '/u/libs/class_max.db'
Loading db file '/u/libs/class_min.db'
Created max/min library relationship:
  Max: /u/libs/class_max.db:class_max
  Min: /u/libs/class_min.db:class_min
```

```
 ...
pt_shell> get_alternative_lib_cells -libraries class_min u_max
{"class_min/OR3", "class_min/OR3P"}
pt_shell> get_alternative_lib_cells -libraries class_max u_min
{"class_max/OR3", "class_max/OR3P"}
pt_shell> get_alternative_lib_cells u_max
{"class_max/OR3P", "class_min/OR3", "class_min/OR3P"}
pt_shell> get_alternative_lib_cells u_min
{"class_max/OR3", "class_max/OR3P", "class_min/OR3P"}
pt_shell> get_alternative_lib_cells u_min -current_library
{"class_min/OR3P"}
pt_shell> get_alternative_lib_cells -base_names u_min
{"OR3", "OR3P"}
```

The following example shows how to find alternative library cells for a lib_cell.

```
pt_shell> get_alternative_lib_cells class_max/OR3
{"class_max/OR3P", "class_min/OR3", "class_min/OR3P"}
pt_shell> get_alternative_lib_cells class_max/OR3 -current_library
{"class_max/OR3P"}
```

SEE ALSO

get_libs(2)
report_alternative_lib_cells(2)
set_min_library(2)
size_cell(2)
link_path(3)
link_path_per_instance(3)

get_app_var

Gets the value of an application variable.

SYNTAX

```
string get_app_var
   [-default | -details | -list]
   [-only_changed_vars]
   var
```

Data Types

```
var     string
```

ARGUMENTS

-default

Gets the default value.

-details

Gets additional variable information.

-list

Returns a list of variables matching the pattern. When this option is used, then the *var* argument is interpreted as a pattern instead of a variable name.

-only_changed_vars

Returns only the variables matching the pattern that are not set to their default values, when specified with **-list**.

var

Specifies the application variable to get.

DESCRIPTION

The **get_app_var** command returns the value of an application variable.

There are four legal forms for this command:

- `get_app_var <var>`
  `Returns the current value of the variable.`

- `get_app_var <var> -default`
  `Returns the default value of the variable.`

- `get_app_var <var> -details`

Returns more detailed information about the variable. See below for details.

- get_app_var -list [-only_changed_vars] <*pattern*>

Returns a list of variables matching the pattern. If *-only_changed_vars* is specified, then only variables that are changed from their default values are returned.

In all cases, if the specified variable is not an application variable, then a Tcl error is returned, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true. See the **sh_allow_tcl_with_set_app_var** man page for details.

When **-details** is specified, the return value is a Tcl list that is suitable as input to the Tcl **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the variable. This key is always present.

value

This key contains the current value of the variable. This key is always present.

default

This key contains the default value of the variable. This key is always present.

help

This key contains the help string for the variable. This key is always present, but sometimes the value is empty.

type

This key contains the type of the application variable. Legal values of for this key are: string, bool, int, real. This key is always present.

constraint

This key describes additional constraints placed on this variable. Legal values for this key are: none, list, range. This key is always present.

min

This key contains the min value of the application variable. This key is present if the constraint is range. The value of this key may be the empty string, in which case the variable only has a max value constraint.

max

This key contains the max value of the application variable. This key is present if the constraint is "range". The value of this key may be the empty string, in which case the variable only has a min value constraint.

list

This key contains the list of legal values for the application variable. This key is present if the constraint is "list".

EXAMPLES

The following are examples of the **get_app_var** command:

```
prompt> get_app_var sh_enable_page_mode
1

prompt> get_app_var sh_enable_page_mode -default
false

foreach {key val} [get_app_var sh_enable_page_mode -details] {   echo "$key: $val"
}
=>  name: sh_enable_page_mode
    value: 1
    default: false
    help: Displays long reports one page at a time
    type: bool
    constraint: none

prompt> get_app_var -list sh_*message
sh_new_variable_message
```

SEE ALSO

report_app_var(2)
set_app_var(2)
write_app_var(2)

get_attribute

Retrieves the value of an attribute on an object or on a collection of objects.

SYNTAX

```
string get_attribute
   [-class class_name]
   [-quiet]
   [-value_list]
   object_spec
   attr_name
```

Data Types

```
class_name       string
object_spec      collection
attr_name        string
```

ARGUMENTS

-class *class_name*

Specifies the class name of the *object_spec* argument, provided the *object_spec* option is a name. The valid values for the *object_spec* option are **design**, **port**, **cell**, **pin**, **net**, **lib**, **lib_cell**, **lib_pin**, and **clock**.

-quiet

Prevents the reporting of messages related to the nonexistence of an object, the nonexistence of an attribute on an object, or the unavailabilty of an attribute on the specified object class. However, more serious usage issues are reported, even when this option is specified.

-value_list

Indicates that the return value should be a list, even if there is only a single object specified to retrieve the attribute. Normally, the return value of the **get_attribute** command is a string if there is only a single object, and a list if multiple objects are specified.

object_spec

Specifies an object from which to retrieve the attribute value. The *object_spec* argument must be either a collection of one or more objects, or a name which is combined with the **-class** option to find the object. If the *object_spec* option is a name, you must also use the **-class** option.

attr_name

Specifies the name of the attribute where the value is retrieved.

DESCRIPTION

This command retrieves the value of an attribute on an object or on a collection of objects. The object is either a string name of exactly one object, or a collection of one or more objects. The return value is a string or a list of strings if multiple objects are specified to retrieve an attribute.

EXAMPLES

In the following example, the first command defines the X attribute for cells. The second command sets the attribute to a value on all cells in this level of the hierarchy. The third command retrieves the value from one cell, combines it with the **full_name** application attribute, and creates a simple report.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
pt_shell> foreach_in_collection sel [get_cells *] {
            echo -n "On '[get_attribute $sel full_name]', "
            echo "X =  [get_attribute $sel X]"
          }
On 'i1', X =  30
On 'i2', X =  30
```

In the next example, the **get_attribute** command is used to retrieve a list of attribute values for a collection of cells.

```
pt_shell> set flipflops [get_cells ff*]
{"ffa", "ffb", "ffc", "ffd"}
pt_shell> get_attribute $flipflops is_sequential
true true true true
```

SEE ALSO

collections(2)
define_user_attribute(2)
foreach_in_collection(2)
list_attributes(2)
remove_user_attribute(2)
report_attribute(2)
set_user_attribute(2)

get_cells

Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command.

SYNTAX

```
collection get_cells
   [-hierarchical]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-filter expression]
   [-of_objects objects]
   patterns
```

Data Types

expression	string
objects	list
patterns	list

ARGUMENTS

-hierarchical

Searches for cells level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder".

-filter *expression*

Filters the collection with the *expression* value. For any cells that match *patterns* or *objects*, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of cells connected to the specified objects. In this case, each object is either a named pin, a pin collection, a named net, a net collection, a named library cell, or a collection of library cells. Note that library cells can not be combined with other types of objects. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

patterns

Matches cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type cell. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

DESCRIPTION

This command creates a collection of cells in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cells match the *patterns* or *objects* option and passes the filter (if specified). If no objects match the criteria, an empty string is returned.

If any *patterns* or *objects* option fails to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_cells** command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the **query_objects** command. In addition, you can assign the **get_cells** result to a variable.

When issued from the command prompt, the **get_cells** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_cells** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command (for example, if you want to display the object class), use the **get_cells** command as an argument to the **query_objects** command.

For information about collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is not. The output is just a display.

```
pt_shell> get_cells "o*" -filter "ref_name == FD2"
{"o_reg1", "o_reg2", "o_reg3", "o_reg4"}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins.

```
pt_shell> set pinsel [get_pins o*/CP]
{"o_reg1/CP", "o_reg2/CP"}

pt_shell> query_objects [get_cells -of_objects $pinsel]
{"o_reg1", "o_reg2"}
```

The following example removes the wire load model from cells i1 and i2.

```
pt_shell> remove_wire_load_model [get_cells {i1 i2}]
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
1
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
link_design(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)

get_clock_network_objects

Returns a collection of objects that belong or relate to the direct clock network.

SYNTAX

```
collection get_clock_network_objects
   -type cell | register | net | pin | clock_gating_output
   [-include_clock_gating_network]
   [clock_list]
```

Data Types

```
clock_list     list
```

ARGUMENTS

-type cell | register | net | pin | clock_gating_output

Specifies one of the following types of clock network objects to return:

- **cell** - Cells that belong to the clock network, including non-inverting or inverting buffers, combinational cells, library defined clock gating cells, etc.
- **register** - Latches, flip-flops or black-box IPs, such as memory cells, that are driven by the clock network.
- **net** - Nets that connect the clock network cells to other clock network cells, to the driven registers, or to the I/O ports of design.
- **pin** - Pins and ports that are connected with the clock network nets. Note that the clock pins of the driven registers are included.
- **clock_gating_output** - Output pins of clock gating cells, either defined by the library, inserted by Power Compiler, automatically inferred by PrimeTime, or manually set by the user. The results are consistent with those of **report_clock_gating_checks**, and can be affected by other clock gating check related commands or variables.

-include_clock_gating_network

Indicates that clock gating networks are included in the clock network.

clock_list

Specifies the clock domains of which the clock network objects are returned. By default, the command returns all clock network objects.

DESCRIPTION

This command returns a collection of clock network objects of certain type (specified by the *object_type* option) that belong or relate to one or several clock domains (specified by the *clock_list* option), or all clock domains if the *clock_list* option is not specified.

A clock network is a special logic part of the design that propagates the clocks from the clock sources to the clock pins of latches, flip-flops (that function as anything but propagating clocks) or black-box IPs. The propagation also stops at design output ports, dangling pins or nets, or the sources of other clocks. The **get_clock_network_objects** command retrieves certain types of objects from the direct clock network (including the latches, flip-flops and black-box IPs driven by the clock network). If the specified clock is a generated clock, the propagation does not go backward to the clock network of its master clock. If the specified clock is a master clock, the propagation does not go forward to the clock network of its generated clocks either.

The **-include_clock_gating_network** option indicates that discrete logic structure functioning as clock gating is taken as belonging to the clock network. Only the typical clock gating logic is considered as qualified clock gating network to be included in the clock network. The typical clock gating logic starts from the output of a level sensitive latch driven by the specified clock, possibly goes through a couple of buffers, and comes back to the specified clock network through one of the input pins of an AND or OR gate. The input pin must be a PrimeTime clock check enable pin, either inferred or manually set. Therefore, similarly, the results can be affected by clock gating check related commands or variables. When the clock gating network is included in the clock network, all objects in the clock gating network is considered as clock network objects. The latch is regarded as a clock network *cell*, but not a clock network *register*. The *clock_gating_output* object type is not affected by this option.

EXAMPLES

The following command returns a collection of clock network pins of all clock domains in the design, not including pins of the clock gating networks.

```
pt_shell> get_clock_network_objects -type pin
```

SEE ALSO

```
create_clock(2)
create_generated_clock(2)
get_clocks(2)
get_generated_clocks(2)
remove_clock(2)
remove_clock_gating_check(2)
remove_disable_clock_gating_check(2)
remove_generated_clock(2)
report_clock(2)
report_clock_gating_check(2)
set_clock_gating_check(2)
set_disable_clock_gating_check(2)
timing_disable_clock_gating_checks(3)
```

get_clocks

Creates a collection of clocks from the current design. You can assign these clocks to a variable or pass them into another command.

SYNTAX

```
collection get_clocks
   [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns
```

Data Types

```
expression      string
patterns        list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-filter *expression*

Filters the collection with the *expression* option. For any clocks that match the *patterns* argument, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result.

patterns

Matches clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_clocks** command creates a collection of clocks in the current design that match certain criteria. The command returns a collection if any clocks match the *patterns* argument and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_clocks** command result to a variable.

When issued from the command prompt, the **get_clocks** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_clocks** command provides a fast, simple way to display clocks in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use the **get_clocks** command as an argument to the **query_objects** option.

For information about collections and the querying of objects, see the **collections** man page. In addition, see the man page for the **all_clocks** command, which also creates a collection of clocks.

EXAMPLES

The following example applies the **set_max_time_borrow** command to all clocks in the design matching "PHI*".

```
pt_shell> set_max_time_borrow 0 [get_clocks "PHI*"]
```

The following example sets the variable clock_list to a collection of clocks that have period of 15.

```
pt_shell> set clock_list [get_clocks * -filter "period==15"]
```

SEE ALSO

```
all_clocks(2)
collections(2)
create_clock(2)
query_objects(2)
report_clock(2)
collection_result_display_limit(3)
```

get_command_option_values

Queries current or default option values.

SYNTAX

```
get_command_option_values
   [-default | -current]

   -command command_name
```

Data Types

```
command_name      string
```

ARGUMENTS

-default

Gets the default option values, if available.

-current

Gets the current option values, if available.

-command *command_name*

Gets the option values for this command.

DESCRIPTION

This command attempts to query a default or current value for each option (of the command) that has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the **-current** or **-default** options is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values is returned as the Tcl result. The even-numbered entries in the list are the names of options that were enabled for default-value-tracking or current-value-tracking and had at least one of these values set to a not-undefined value). Each odd-numbered entry in the list is the default or current value of the option name preceding it in the list.

Any options that were not enabled for either default-value-tracking nor current-value-tracking are omitted from the output list. Similarly, options that were enabled for default-value-tracking or current-value-tracking, but for

which no (not-undefined) default or current value is set, are omitted from the result list.

If neither **-current** nor **-default** is specified, then for each command option that has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is as follows:

- The current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set;
- Otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set;
- Otherwise the name and value pair for the option is not included in the result list.

If **-current** is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name and value pair for the option is omitted from the result list.

If **-default** is specified, the value returned for an option is the default value if default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name and value pair for the option is omitted from the result list.

The result list from **get_command_option_values** includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command issues a Tcl error in a variety of situations, such as if an invalid command name was passed in with **-command**.

EXAMPLES

The following example shows the use of **get_command_option_values**:

```
prompt> test -opt1 10 -opt2 20
1

prompt> get_command_option_values -command test
-bar1 10 -bar2 20
```

SEE ALSO

preview(2)
set_command_option_value(2)

get_core_area

Creates a collection containing the core area of the current design.

SYNTAX

```
collection get_core_area
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command returns a collection containing the core area of the current design. If the core area is not defined, the command returns an empty string.

The core area is a box that contains all rows. It represents the placeable area for standard cells. The core area is typically smaller than the cell boundary. Pads, I/O pins, and top-level power and ground rings are typically found outside the core area. Standard cells, macros, and wire tracks are typically found inside the core area. There should be only one core area in a design.

The core_area object is a first class object,and it has multiple attributes, such as bbox, cell_id, name and so on.You can use **list_attributes** and **report_attribute** to browse attributes.You can use **get_attribute** and **lindex** to get coordinates of the current design die area.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the core area of the current design.

```
prompt> get_core_area
{unit_core}
```

The following example shows how to list the attributes of the core_area class by using **list_attributes** command.

```
prompt> list_attributes -application -class core_area
```

The following example shows how to get the attribute values of the core_area object by using **report_attribute** command.

```
prompt> report_attribute -application -class core_area [get_core_area]
```

The following example shows how to get coordinates of the current design die area,by using **get_attribute** command.

```
prompt> get_attribute [get_core_area] bbox
{0.000 0.000} {300.480 396.620}
```

SEE ALSO

get_die_area(2)
get_attribute(2)
list_attributes(2)

get_current_power_domain

Gets the power domains that are included in the power analysis.

This command works only in UPF mode.

SYNTAX

```
string get_current_power_domain
```

ARGUMENTS

None.

DESCRIPTION

The **set_current_power_domain** command provides the control of calculating power consumption for the domains of interest. Domain-based power consumption data can be generated for a multipower domain design. Only the power dissipated in the blocks covered by the current power domain list is calculated and reported. The **get_current_power_domain** command returns the power domains included in the power analysis.

EXAMPLES

The following example shows generating domain-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by power domains PD1 and PD2, respectively. The first **report_power** command generates the power analysis results for the whole design, which is the default behavior. The second **report_power** command generates the power analysis results for the power consumed in power domain PD1. The third **report_power** command generates the results for the power consumed in power domain PD2.

```
pt_shell> ...
pt_shell> create_power_domain PD1 -elements {InstA}
pt_shell> create_power_domain PD2 -elements {InstB}
pt_shell> ...
pt_shell> report_power
pt_shell> set_current_power_domain PD1
pt_shell> report_power
pt_shell> get_current_power_domain
pt_shell> set_current_power_domain PD2
pt_shell> report_power
```

```
pt_shell> get_current_power_domain
```

SEE ALSO

set_current_power_domain(2)

get_current_power_net

Gets the power nets that are included in the power analysis.

This command works in UPF mode only.

SYNTAX

```
string get_current_power_net
```

ARGUMENTS

None.

DESCRIPTION

The *set_current_power_net* command provides control of calculating power consumption for the power nets of interest. Power net-based power consumption data can be generated for a multi-supply design. Only the power dissipated on the cell or part of cell that is supplied by the current power netlist is calculated and reported. The **get_current_power_net** command returns the power nets being included in the power analysis.

EXAMPLES

The following example shows generating power net-based power analysis results. The design has two subblocks "InstA" and "InstB", which is covered by power domain PD1 and PD2, respectively. There are a total of six supply nets defined. Among them, four are power nets and two are ground nets. The primary power net for power domain "PD1" is "VDDIS", and the primary power net for power domain "PD2" is "VDDGS". The first **report_power** command generates the power analysis results for the whole design, which is the default behavior. The second **report_power** command generates the power consumption associated with power net "VDDI". The third **report_power** command generates the power consumption associated with power net "VDDIS", which is the output of power switch "top_header". The fourth and fifth reports are for "VDDG" and "VDDGS", respectively.

```
pt_shell> ...
pt_shell> create_power_domain PD1 -elements {InstA}
pt_shell> create_power_domain PD2 -elements {InstB}
pt_shell> create_supply_net VDDI  -domain PD1
pt_shell> create_supply_net VDDIS -domain PD1
pt_shell> create_supply_net VSS   -domain PD1
pt_shell> set_domain_supply_net PD1 -primary_power_net VDDIS -primary_ground_net VSS
```

```
pt_shell> connect_supply_net -ports InstA/LS_u1/VDDL VDDI
pt_shell> connect_supply_net -ports InstA/LS_u1/VDD VDDIS
pt_shell> connect_supply_net -ports InstA/LS_u1/VSS VSS
pt_shell> create_power_switch top_header \
                     -domain PD1 \
                     -output_supply_port {NSLEEPOUT VDDIS} \
                     -input_supply_port {NSLEEPIN VDDI} \
                     -on_state {state1 NSLEEPIN {CNTL}} \
                     -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> create_supply_net VDDG  -domain PD2
pt_shell> create_supply_net VDDGS -domain PD2
pt_shell> create_supply_net VSS   -domain PD2 -reuse
pt_shell> set_domain_supply_net PD2 -primary_power_net VDDGS -primary_ground_net VSS
pt_shell> create_power_switch gprs_header \
                     -domain PD2 \
                     -output_supply_port {NSLEEPOUT VDDGS} \
                     -input_supply_port {NSLEEPIN VDDG} \
                     -on_state {state1 NSLEEPIN {CNTL}} \
                     -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> report_power
pt_shell> set_current_power_net { VDDI }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDIS }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDG }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDGS }
pt_shell> report_power
pt_shell> get_current_power_net
```

SEE ALSO

set_current_power_net(2)

get_defined_commands

Get information on defined commands and groups.

SYNTAX

```
string get_defined_commands [-details]
   [-groups]
   [pattern]

string pattern
```

ARGUMENTS

-details

Get detailed information on specific command or group.

-groups

Search groups rather than commands

pattern

Return commands or groups matching pattern. The default value of this argument is "*".

DESCRIPTION

The **get_defined_commands** gets information about defined commands and command groups. By default the command returns a list of commands that match the specified pattern.

When **-details** is specified, the return value is a list that is suitable as input to the **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key name, and each even-numbered element in the list os the value of the previous key. The **-details** option is only legal if the pattern matches exactly one command or group.

When **-group** is specified with **-details**, the supported keys are as follows:

name

This key contains the name of the group.

info

This key contains the short help for the group.

commands

This key contains the commands in the group

When **-details** is used with a command, the supported keys are as follows:

name

This key contains the name of the command.

info

This key contains the short help for the command.

groups

This key contains the group names that this command belongs to.

options

This key contains the options defined for the command. The value is a list.

return

This key contains the return type for the command.

Each element in the *options* list also follows the key value pattern. The set of available keys for options are as follows:

name

This key contains the name of the option.

info

This key contains the short help for the option.

value_info

This key contains the short help string for the value

type

The type of the option.

required

The value will be 0 or 1 depending if the option is optional or required.

is_list

Will be 1 if the option requires a list.

list_length

If a list contains the list length constraint. One of any, even, odd, non_empty or a number of elements.

allowed_values

The allowed values if the option has specified allowed values.

min_value

The minimum allowed value if the option has specified one.

max_value

The maximum allowed value if the option has specified one.

EXAMPLES

```
prompt> get_defined_commands *collection
add_to_collection append_to_collection copy_collection filter_collection
foreach_in_collection index_collection sort_collection
prompt> get_defined_commands -details sort_collection
name sort_collection info {Create a sorted copy of the collection}
groups {} options {{name -descending info {Sort in descending order}
value_info {} type boolean required 1 is_list 0} {name -dictionary info
{Sort strings dictionary order.} value_info {} type boolean required 1
is_list 0} {name collection info {Collection to sort} value_info
collection type string required 0 is_list 0} {name criteria info {Sort
criteria - list of attributes} value_info criteria type list required 0
is_list 1 list_length non_empty}}
```

SEE ALSO

```
help(2)
man(2)
```

get_designs

Creates a collection of one or more designs loaded into PrimeTime. You can assign these designs to a variable or pass them into another command.

SYNTAX

```
collection get_designs
   [-hierarchical]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-filter expression]
   patterns
```

Data Types

expression	string
patterns	list

ARGUMENTS

-hierarchical

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. Using this option does not force an autolink.

-filter *expression*

Filters the collection with the *expression* value. For any designs that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

patterns

Matches design names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type design.

DESCRIPTION

This command creates a collection of designs from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any designs match the *patterns* option and pass the filter, if specified. If no objects matched your criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_designs** command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the **query_objects** command. In addition, you can assign the **get_designs** result to a variable.

When issued from the command prompt, the **get_designs** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable .

The "implicit query" property of the **get_designs** command provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the **query_objects** command (for example, if you want to display the object class), use the **get_designs** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the designs that begin with 'mpu.' Although the output looks like a list, it is just a display. A complete listing of designs is available using the **list_designs** command.

```
pt_shell> get_designs mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

The following example shows that, given a collection of designs, you can remove those designs.

```
pt_shell> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

SEE ALSO

```
collections(2)
filter_collection(2)
list_designs(2)
query_objects(2)
regexp(2)
remove_design(2)
collection_result_display_limit(3)
```

get_die_area

.prod icc Creates a collection containing the die area of the current design. .prod all

SYNTAX

```
.prod icc
collection get_die_area
.prod all
```

ARGUMENTS

.prod icc

This command has no arguments. .prod all

DESCRIPTION

.prod icc

This command returns a collection containing the die area of the current design. If the die area is not defined, the command returns an empty string.

The die area is also known as the cell boundary. It represents the silicon boundary of a chip and typically encloses all objects of a design, such as pads, I/O pins, cells, and so on. There should be only one die area in a design.

The die area object has multiple attributes, such as bbox, cell_id, name, and so on. You can use the **list_attributes** and **report_attribute** commands to browse attributes. You can use the **get_attribute** command and **lindex** to get coordinates of the current design die area. .prod all

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

.prod icc

The following example gets the die area of the current design.

```
prompt> get_die_area
.prod icc
{top_die}

.prod all
```

.prod icc

The following example shows how to use the **list_attributes** command to list the attributes of the die_area class.

```
prompt> list_attributes -application -class die_area
```

The following example shows how to use the **report_attribute** command to get the attribute values of the die_area object.

```
prompt> report_attribute -application\
-class die_area [get_die_area]
```

The following example shows how to use the **get_attribute** command to get coordinates of the current design die area.

```
prompt> get_attribute [get_die_area] bbox
{0.000 0.000} {300.480 396.620}
```

.prod all

SEE ALSO

.prod icc
get_attribute(2)
get_core_area(2)
list_attributes(2)
.prod all

get_em_max_toggle_rate

Reports electromigration (EM) max toggle rate data from library for a specified instance pin or library cell pin .

SYNTAX

```
status get_em_max_toggle_rate
   [-related_pin pin]
   [-transition_time transition_time]
   [-transition_type transition_type]
   [-output_load load]
   [-current_type type]
   [-verbose]
   object_name
```

Data Types

pin	pin
transition_time	float
transition_type	max|min
load	float
type	string
object_name	string

ARGUMENTS

-related_pin *pin*

Reports max EM toggle rate for the specified arc.

-transition_time *float*

Specifies the transition time at input.

-transition_type *max|min*

Specifies the transition type at input. Valid values are max and min.

-output_load *float*

Specifies the load at output pin.

-current_type *type*

Specifies the table to use for toggle rate calculations. Valid values are average , rms and peak. If the switch is not specified, the command checks only against RMS table by default.

-verbose

Shows the slew rate and cap values used for max toggle rate violation calculations. Also reports errors for missing EM data for pins.

object_name

Specifies an instance pin from the current design or a library cell pin.

DESCRIPTION

This command reports the EM max toggle rate from library for a specific cell pin or library cell pin.

EXAMPLES

The following command reports the max toggle rate for the specified pin 'cell_em/test_em_cell_2/Z' with a specified load.

```
pt_shell> get_em_max_toggle_rate -output_load 5.4 cell_em/test_em_cell_2/Z
```

The following command reports the max toggle rate for the pin 'cell_em/test_em_cell_2/A' with a specified transition time .

```
pt_shell> get_em_max_toggle_rate  -transition_time 0.5  cell_em/test_em_cell_2/A
```

```
The following command reports EM max toggle rate arc for a specified arc.
pt_shell> get_em_max_toggle_rate -transition_time 2.053 -output_load 2.053  -related_pin cell_em/test_e
```

The following command reports max toggle for the specfied pin from average type table.

```
pt_shell> get_em_max_toggle_rate -current_type ave cell_em/test_em_cell/Z
```

SEE ALSO

```
report_cell_em_violation(2)
get_switching_activity(2)
```

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

```
collection get_generated_clocks
   [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns
```

Data Types

```
expression       string
patterns         list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-filter *expression*

Filters the collection with the *expression* option. For any generated clocks that match the *patterns* option, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

patterns

Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_generated_clocks** command creates a collection of generated clocks from the current design that match certain criteria. The command returns a collection if any generated clocks match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the **create_generated_clock** command. To remove a generated clock from the design, use the **remove_generated_clock** command. To show information about clocks and generated clocks in the design, use the **report_clock** command.

You can use the **get_generated_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_generated_clocks** command result to a variable.

When issued from the command prompt, the **get_generated_clocks** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_generated_clocks** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command, use the **get_generated_clocks** command as an argument to the **query_objects** command. For example, use this to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following command applies the **set_clock_latency** command on all generated clocks in the design matching the "GEN*".

```
pt_shell> set_clock_latency 2.0 [get_generated_clocks "GEN*"]
```

The following command removes all the generated clocks in the design matching "GEN1*".

```
pt_shell> remove_generated_clock [get_generated_clocks "GEN1*"]
```

SEE ALSO

```
collections(2)
create_generated_clock(2)
query_objects(2)
remove_generated_clock(2)
report_clock(2)
collection_result_display_limit(3)
```

get_latch_loop_groups

Returns a list of collections of pins, each collection containing the transparent latch data pins contained within one latch loop group.

SYNTAX

```
list get_latch_loop_groups
   [-of_objects pin_list]
   [-loop_breakers_only]
```

Data Types

pin_list	list

ARGUMENTS

-of_objects *pin_list*

Specifies a collection of data pins of transparent latches. This option returns only pins of loop groups containing the specified pins. By default, the command returns one collection for each loop group in the design.

-loop_breakers_only

Specifies that the returned collection contains only pins that are loop breaker latch data pins (that have a **true** value for the **is_latch_loop_breaker** pin attribute). By default, all transparent latch data pins within the loop group are returned in the collections.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all latch data pins of intersecting loops is formed. The **get_latch_loop_groups** command analyzes the design and returns a collection for each loop group formed. The collection contains the latch data pins of the loop group. Combinational pins are not included in the results.

You can use the results of the **get_latch_loop_groups** command to find paths within loops that violate timing. For an example, see the EXAMPLES section.

Before using the **get_latch_loop_groups** command, you must set the **timing_enable_through_paths** variable to **true**.

EXAMPLES

The following example reports violating paths within latch loops by using the **get_latch_loop_groups** command. No paths outside the loops are reported.

```
pt_shell> foreach a_loop_group [get_latch_loop_groups -loop_breakers_only] {
            report_timing -from $a_loop_group -to $a_loop_group -slack_lesser_than 0.0
          }
```

SEE ALSO

report_latch_loop_groups(2)
timing_enable_through_paths(3)

get_layers

Creates a collection of one or more layers.

SYNTAX

```
collection get_layers
   [-quiet]
   [-regexp | -exact]
   [-nocase]
   [-filter expression]
   [-include_system]
   [patterns]
```

Data Types

```
expression   string
patterns     string
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each layer in the collection, the expression is evaluated based on the layer's attributes. If the expression evaluates to true, the layer is included in the result.

To see the list of layer attributes that you can use in the expression, use the **list_attributes -application -class layer** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-include_system

Includes the system layers in the collection. Without this option, only user layers that are defined in the technology file can be included in the collection. Layer numbers from 1 to 187 are for user layers, and the layer numbers great than 187 are for system layers.

patterns

Creates a collection of layers whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

If you do not specify this argument, the command uses the * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of layers defined in the technology file for the current library that match the specified criteria. If you use the **-include_system** option, the collection can also contain system layers.

The command returns a collection if any layers match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns a layer for the name M1R.

```
prompt> get_layers M1R
{M1R}
```

The following example returns layers for routing.

```
prompt> get_layers -filter is_routing_layer==true
{METAL5 METAL METAL2 METAL3 METAL4 M6 M7 M8 PA}
```

SEE ALSO

collections(2)
filter_collection(2)
get_layer_attribute(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_lib_cells

Creates a collection of library cells from libraries loaded into PrimeTime. You can assign these library cells to a variable or pass them into another command.

SYNTAX

```
collection get_lib_cells
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-of_objects objects]
   patterns
```

Data Types

expression	string
objects	list
patterns	list

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* value. For any library cells that match the *patterns* or *objects* argument, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the **-regexp** option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

-of_objects ***objects***

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. In this case, each object is either a named library pin or netlist cell, or a library pin collection or a netlist cell collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

patterns

Matches library cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib_cell. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

DESCRIPTION

This command creates a collection of library cells from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cells match the *patterns* or *objects* option and pass the filter, if specified. If no objects match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_lib_cells** command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the **query_objects** command. In addition, you can assign the **get_lib_cells** command result to a variable.

When issued from the command prompt, the **get_lib_cells** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_lib_cells** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command (for example, if you want to display the object class), use the **get_lib_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all library cells that are in the misc_cmos library and begin with AN2. Although the output looks like a list, it is just a display.

```
pt_shell> get_lib_cells misc_cmos/AN2*
{"misc_cmos/AN2", "misc_cmos/AN2P"}
```

The following example shows one way to find out the library cell used by a particular cell.

```
pt_shell> get_lib_cells -of_objects [get_cells o_reg1]
{"misc_cmos/FD2"}
```

SEE ALSO

```
collections(2)
filter_collection(2)
get_cells(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_lib_pins

Creates a collection of library cell pins from libraries loaded into PrimeTime. You can assign these library cell pins to a variable or pass them into another command.

SYNTAX

```
collection get_lib_pins
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [patterns | -of_objects objects]
```

Data Types

```
expression      string
patterns        list
objects         list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* option. For any library cell pins that match the *patterns* option (or the *objects* option), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* option and the *-exact* option are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

patterns

Matches library cell pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. The *patterns* option can also include collections of type **lib_pin**. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

-of_objects *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell or netlist pin, or a library cell collection or netlist pin collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_lib_pins** command creates a collection of library cell pins from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cell pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the **regexp** Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_lib_pins** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_lib_pins** command result to a variable.

When issued from the command prompt, the **get_lib_pins** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_lib_pins** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command, use the **get_lib_pins** command as an argument to the **query_objects** command. For example, use this if you want to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
pt_shell> [get_lib_pins misc_cmos/AN2/*
{"misc_cmos/AN2/A", "misc_cmos/AN2/B", "misc_cmos/AN2/Z"}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist.

```
pt_shell> get_lib_pins -of_objects o_reg1/Q
{"misc_cmos/FD2/Q"}
```

SEE ALSO

```
collections(2)
filter_collection(2)
get_libs(2)
get_lib_cells(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_lib_timing_arcs

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing.

SYNTAX

```
string get_lib_timing_arcs
   [-to to_list]
   [-from from_list]
   [-of_objects object_list]
   [-filter expression]
   [-quiet]
```

Data Types

to_list	list
from_list	list
object_list	list
expression	string

ARGUMENTS

-to *to_list*

Specifies the "to" library pins, or ports. All backward library arcs from the specified library pins or ports are considered.

-from *from_list*

Specifies the "from" library pins, or ports. All forward library arcs from the specified library pins or ports are considered.

-of_objects *object_list*

Specifies library cells or timing arcs. If a library cell is specified, all library cell arcs of that cell are considered. If a timing arc collection is given in the object list, the corresponding library timing arc is considered.

-filter *expression*

Specifies the filter expression. A filter expression is a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of library arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value, by means of an operator.

-quiet

Specifies that all messages are to be suppressed.

DESCRIPTION

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing. Use the **foreach_in_collection** command to iterate among the library arcs in the collection. You can use the **get_attribute** command to obtain information about the paths. You can also use the filter expression to filter the library arcs that satisfy the specified conditions.

The following attributes are supported on library timing arcs:

```
from_lib_pin
is_disabled
is_user_disabled
mode
object_class
sdf_cond
sense
to_lib_pin
```

One attribute of a library timing arc is the **from_lib_pin**, which is the library pin from which the timing arc begins. In the same way, **to_lib_pin** is the library pin to which the timing arc ends. To get more information on **from_lib_pin** and **to_lib_pin**, use the **get_attributes** command. The **object_class** attribute holds the class name of library timing arcs: **lib_timing_arc**.

See the **collections** and **foreach_in_collection** man pages for more information.

EXAMPLES

The following procedure is an example you can use of how the collection returned from an invocation of **get_lib_timing_arcs** can be used to provide useful and readable cell level arc information.

```
proc show_lib_arcs {args} {
  set lib_arcs [eval [concat get_lib_timing_arcs $args]]
  echo [format "%15s    %-15s  %18s" "from_lib_pin" "to_lib_pin" \
            "sense"]
  echo [format "%15s    %-15s  %18s" "------------" "----------" \
            "-----"]

  foreach_in_collection lib_arc $lib_arcs {
    set fpin [get_attribute $lib_arc from_lib_pin]
    set tpin [get_attribute $lib_arc to_lib_pin]
    set sense [get_attribute $lib_arc sense]
    set from_lib_pin_name [get_attribute $fpin base_name]
    set to_lib_pin_name [get_attribute $tpin base_name]
    echo [format "%15s -> %-15s  %18s" \
        $from_lib_pin_name $to_lib_pin_name \
        $sense]
  }
}
```

```
pt_shell> show_lib_arc -of_objects [get_timing_arcs -of_objects ffa]

   from_lib_pin    to_lib_pin                    sense
   ------------    ----------                    -----
             CP -> D                    setup_clk_rise
             CP -> D                     hold_clk_rise
```

```
              CP -> Q                        rising_edge
              CP -> QN                       rising_edge
              CD -> Q                          clear_low
              CD -> QN                        preset_low
```

```
pt_shell> show_lib_arc -of_objects mylib/AN2

   from_lib_pin    to_lib_pin                     sense
   ------------    ----------                     -----
              A -> Z                     positive_unate
              B -> Z                     positive_unate
```

```
pt_shell> show_lib_arc -from mylib/AN2/A

   from_lib_pin    to_lib_pin                     sense
   ------------    ----------                     -----
              A -> Z                     positive_unate
```

SEE ALSO

collections(2)
filter_collection(2)
foreach_in_collection(2)
get_attribute(2)
get_timing_arcs(2)

get_libs

Creates a collection of libraries loaded into PrimeTime. You can assign these libraries to a variable or pass them into another command.

SYNTAX

```
collection get_libs
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-of_objects objects]
   [patterns]
```

Data Types

```
expression       string
objects          list
patterns         list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* option. For any libraries that match *patterns* (or *objects*), the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. the *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

patterns

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **get_libs** command creates a collection of libraries from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any libraries match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the **regexp** Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_libs** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_libs** command result to a variable.

When issued from the command prompt, the **get_libs** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_libs** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command, use the **get_libs** command as an argument to the **query_objects** command. For example, use this if you want to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all loaded libraries. Although the output looks like a list, it is just a display. Note that a complete listing of libraries is available using the **list_libs** command.

```
pt_shell> get_libs *
{"misc_cmos", "misc_cmos_io"}
```

The following example shows that given a library collection, you can remove those libraries. Note that you cannot remove libraries if they are referenced by a design.

```
pt_shell> remove_lib [get_libs misc*]
Removing library misc_cmos...
Removing library misc_cmos_io...
```

SEE ALSO

```
collections(2)
filter_collection(2)
list_libs(2)
query_objects(2)
regexp(2)
remove_lib(2)
collection_result_display_limit(3)
```

get_license

Obtains a license for a feature.

SYNTAX

```
status get_license
   [-quantity num_licenses]
   feature_list
```

Data Types

num_licenses	integer
feature_list	list

ARGUMENTS

-quantity *num_licenses*

Specifies the total number of licenses to check out for each feature. If a license is already checked out, the command acquires only the additional licenses needed to bring the total to the specified quantity. If you do not use this option, the command checks out one license per feature.

This option is useful if you need to reserve multiple licenses for a multicore run. You can reserve the required number of licenses early in the session, rather than risking a failure or a performance degradation later in the session if the licenses are not available.

feature_list

Specifies a list of features to obtain. To specify more than one feature, enclose the list enclosed in curly braces ({ }).

DESCRIPTION

This command obtains licenses for the specified features. These features remain checked out by you until you run the **remove_license** command or until you exit the program.

To list the features that you currently have checked out, run the **list_licenses** command.

EXAMPLES

This example obtains two licenses of the PrimeTime PX feature:

```
pt_shell> get_license PrimeTime-PX -quantity 2
Information: Checked out license 'PrimeTime-PX' (PT-019)
Information: Checked out license 'PrimeTime-PX' (PT-019)
1
```

SEE ALSO

```
license_users(2)
list_licenses(2)
remove_license(2)
```

get_macro_current_sources

Creates a collection of macro current sources that match the specified criteria.

SYNTAX

```
collection get_macro_current_sources
   [-within region]
   [-filter expression]
   [patterns]
```

Data Types

```
region        list of points
expression    string
patterns      list
```

ARGUMENTS

-filter *expression*

Filters the collection with the specified expression. For each macro current source in the collection, the expression is evaluated based on the macro current source's attributes. If the expression evaluates to true, the macro current source is included in the result.

To see the list of macro current source attributes that you can use in the expression, use the **list_attributes -application -class macro_current_source** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all macro current sources that are completely inside the specified region and do not overlap the boundary. The region boundary can only be a rectangle.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The coordinate unit is specified in the technology file; typically the unit is microns.

If you do not specify the **-within** option, the command searches the entire design.

patterns

Creates a collection of macro current sources whose names match the specified patterns. Macro current sources are named based on the CONN view top level macro cell they reside in and their index number:

- A macro current source object may be named _mm_dev_cel_inst_VDD/*n*

The *n* value in the macro current source name is an integer value.

Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive.

If you do not specify this argument, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of macro current sources by selecting macro current sources from the current design that match the specified criteria.

This command does not support searching across physical hierarchy, this means the blocks in the design are not searched beyond the top level.

The command returns a collection if any macro current sources match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

This command works only on designs opened as CONN view.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the macro current sources in a specified area. Although the output looks like a list, it is only a display.

```
prompt> get_macro_current_sources -within {{102 15} {103 16}}
{"_mm_dev_cel_inst_VDD/11827", "_mm_dev_cel_inst_VDD/11828", "_mm_dev_cel_inst_VSS/19452"}
```

The following example queries the macro current sources that have the specified layer number.

```
prompt> get_macro_current_sources \
   -filter {layer_number==15}
{"_mm_dev_cel_inst_VDD/11827", "_mm_dev_cel_inst_VDD/11828", "_mm_dev_cel_inst_VDD/11829"...}
```

The following examples get the macro current sources inside the CONN view top level macro cell referred by _mm_dev_cel_inst_VDD.

```
prompt> get_macro_current_sources _mm_dev_cel_inst_VDD/11827
{"_mm_dev_cel_inst_VDD/11827"}

prompt> get_macro_current_sources _mm_dev_cel_inst_VDD/*
```

```
{"_mm_dev_cel_inst_VDD/11827", "_mm_dev_cel_inst_VDD/11828", "_mm_dev_cel_inst_VDD/11829"...}
```

SEE ALSO

collections(2)
filter_collection(2)
list_attributes(2)
wildcards(3)

get_macro_mtcmos

Creates a collection of macro_mtcmos objects that match the specified criteria.

SYNTAX

```
collection get_macro_mtcmoss
   [-within region]
   [-filter expression]
   [patterns]
```

Data Types

```
region        list of points
expression    string
patterns      list
```

ARGUMENTS

-filter *expression*

Filters the collection with the specified expression. For each macro_mtcmos object in the collection, the expression is evaluated based on the macro_mtcmos object's attributes. If the expression evaluates to true, the macro_mtcmos object is included in the result.

To see the list of macro_mtcmos object attributes that you can use in the expression, use the **list_attributes -application -class macro_mtcmos** command.

For more information about how to use the -filter option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all macro_mtcmos objects that are completely inside the specified region and do not overlap the boundary. The region boundary can only be a rectangle.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The coordinate unit is the one specified in the technology file. Usually the unit is microns.

If you do not specify the -within option, the command searches the entire design.

patterns

Creates a collection of macro_mtcmos objects whose names match the specified patterns. Macro current sources are named based on the CONN view top-level macro cell where they reside and their index number.

Note that

- A macro_mtcmos object object might be named hm_mtcmos_*n*

where the *n* value is an integer.

Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive.

If you do not specify this argument, the command uses the * (asterisk) character as the default pattern.

DESCRIPTION

This command creates a collection of macro_mtcmos objects by selecting macro_mtcmos objects from the current design that match the specified criteria.

This command does not support searching across physical hierarchy, which means the blocks in the design are not searched beyond the top level.

The command returns a collection if any macro_mtcmos objects match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and object querying, see the **collections** man page.

This command works only on designs that are opened as CONN views.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the macro_mtcmos objects in a specified area. Although the output looks like a list, it is only a display.

```
prompt> get_macro_mtcmoss -within {{102 15} {103 16}}
{"hm_mtcmos_11827", "hm_mtcmos_11828", "hm_mtcmos_19452"}
```

The following example queries the macro_mtcmos objects that have the specified layer number.

```
prompt> get_macro_mtcmoss -filter {layer_number==15}
{"hm_mtcmos_11827", "hm_mtcmos_11828", "hm_mtcmos_11829"...}
```

The following examples get the macro_mtcmos objects inside the CONN view top-level macro cell referred by macro_mtcmos object names.

```
prompt> get_macro_mtcmoss hm_mtcmos_11827
{"hm_mtcmos_11827"}
```

```
prompt> get_macro_mtcmoss hm_mtcmos_*
{"hm_mtcmos_0", "hm_mtcmos_1", "hm_mtcmos_2"...}
```

SEE ALSO

collections(2)
filter_collection(2)
list_attributes(2)
wildcards(3)

get_message_ids

Get application message ids

SYNTAX

```
string get_message_ids [-type severity]
[pattern]

string severity
string pattern
```

ARGUMENTS

-type *severity*

Filter ids based on type (Values: Info, Warning, Error, Severe, Fatal)

pattern

Get IDs matching pattern (default: *)

DESCRIPTION

The **get_message_ids** command retrieves the error, warning and informational messages used by the application. The result of this command is a Tcl formatted list of all message ids. Information about the id can be queried with the **get_message_info** command.

EXAMPLES

The following code finds all error messages and makes the application stop script execution when one of these messages is encountered.

```
foreach id [get_message_ids -type Error] {
  set_message_info -stop_on -id $id
}
```

SEE ALSO

```
print_message_info(2)
set_message_info(2)
suppress_message(2)
```

get_message_info

Returns information about diagnostic messages.

SYNTAX

```
integer get_message_info
   [-error_count | -warning_count | -info_count
     | -limit l_id | -occurrences o_id | -suppressed s_id | -id i_id]
```

Data Types

```
l_id      string
o_id      string
s_id      string
i_id        string
```

ARGUMENTS

-error_count

Returns the number of error messages issued so far.

-warning_count

Returns the number of warning messages issued so far.

-info_count

Returns the number of informational messages issued so far.

-limit *l_id*

Returns the current user-specified limit for a given message ID. The limit was set with the **set_message_info** command.

-occurrences *o_id*

Returns the number of occurrences of a given message ID.

-suppressed *s_id*

Returns the number of times a message was suppressed either using **suppress_message** or due to exceeding a user-specified limit.

-id *i_id*

Returns information about the specified message. The information is returned as a Tcl list compatible with the array set command.

DESCRIPTION

The **get_message_info** command retrieves information about error, warning, and informational messages. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to retrieve recorded information about generated diagnostic messages. For example, you can stop a script after a certain number of errors have occurred, or monitor the number of messages issued by a single command.

You can also find out how many times a specific message has occurred, or how many times it has been suppressed. Also, you can find out if a limit has been set for a particular message ID.

EXAMPLES

The following example uses the **get_message_info** command to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount:

```
prompt> proc \
  do_command {limit} {
    set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {
      return -code error "Too many errors"
    }
    ...
  }
```

The following example uses the **get_message_info** command to retrieve information on the CMD-014 message:

```
prompt> get_message_info -id CMD-014
id CMD-014 severity Error limit 0 occurrences 0 suppressed 0 message
{Invalid %s value '%s' in list.}
```

SEE ALSO

print_message_info(2)
set_message_info(2)
suppress_message(2)
get_message_ids(2)

get_net_shapes

Creates a collection of net shapes that match the selection criteria.

SYNTAX

```
collection get_net_shapes
   [-within region
     | -touching region
     | -intersect region
     | -at point]
   [-filter expression]
   [-quiet]
   [-type vw | hw | path | poly]
   [patterns
     | -of_objects supply_nets]
```

Data Types

```
region        list
point         string
expression    string
patterns      list
supply_nets   collection
```

ARGUMENTS

-within *region*

Creates a collection that contains all net shapes that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for net shapes in the entire design.

-touching *region*

Creates a collection that contains all net shapes that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for net shapes in the entire design.

-intersect *region*

Creates a collection that contains all net shapes that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be rectangle or polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one.

-at *point*

Creates a collection that contains all net shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for net shapes in the entire design.

-filter *expression*

Filters the collection with the specified expression.

For each net shape in the collection, the expression is evaluated based on the net shape's attributes. If the expression evaluates to true, the net shape is included in the result.

To see the list of net shape attributes that you can use in the expression, use the **list_attributes -application -class shape** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-quiet

Suppresses the reporting of warning and information messages.

-type vw | hw | path | poly

Specifies the type of net shape, where **vw** is for vertical wire, **hw** is for horizontal wire, **path** is the actual path, and **poly** is for polygon.

patterns

Creates a collection of net shapes whose names match the specified patterns. For net shapes associated with horizontal wires, the net shape name starts with HWIRE#. For net shapes associated with vertical wires, the net shape name starts with VWIRE#. For net shapes associated with paths, the net shape name starts with PATH#. For net shapes associated with polygons, the net shape name starts with POLYGON#. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *supply_nets*

Creates a collection of net shapes connected to the specified supply nets.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of net shapes by selecting net shapes from the current design that match the specified criteria.

The command returns a collection if any net shapes match the criteria. If no objects match the criteria, the command returns an empty string.

When using the **-within**, **-touching**, or **-intersect** option, you can specify the region boundary as a rectangle or polygon. The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle. The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} … {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

This command does not support searching across physical hierarchy, this means the blocks in the design are not searched beyond the top level.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all net shapes within the specified rectangle.

```
prompt> get_net_shapes * -within {{4 20} {15 30}}
{HWIRE#7264 VWIRE#6766}
```

The following example gets the net shapes that intersect with the specified polygon:

```
prompt>get_net_shapes \
   -intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{HWIRE#39936}
```

The following example gets the net shapes associated with the VDD supply net that intersect with the specified rectangle:

```
prompt> get_net_shapes -of_objects [get_supply_nets VDD] \
   -intersect {{4 20} {15 30}}
{VWIRE#6766}
```

The following example gets the net shapes with a **route_type** attribute of "P/G Std. Cell Pin Conn":

```
prompt> get_net_shapes \
   -filter {route_type=="pg_std_cell_pin_conn"}
{PATH#1134336 PATH#1134337 PATH#1134338 PATH#1134339 PATH#1134340}
```

SEE ALSO

```
get_supply_nets(2)
collections(2)
get_user_shapes(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)
```

get_nets

Creates a collection of nets from the netlist. You can assign these nets to a variable or pass them into another command.

SYNTAX

```
collection get_nets
   [-hierarchical]
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-top_net_of_hierarchical_group]
   [-segments]
   [-boundary_type btype]
   [-of_objects objects]
   patterns
```

Data Types

expression	string
btype	string
objects	list
patterns	list

ARGUMENTS

-hierarchical

Searches for nets level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a net block1/muxsel, a hierarchical search would find it using "muxsel." You cannot use the *-hierarchical* option with the *-of_objects* option.

-filter *expression*

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard

patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-exact* option, makes matches case-insensitive. You can use *-nocase* only when you also use the *-exact* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

-top_net_of_hierarchical_group

Keep only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. Although this option can be used when there are multiple nets specified, it is best used in combination with the *-segments* option and with a single net.

-segments

Return all global segments for given nets. This modifies the initial search which matches *patterns* or *objects*. It is applied after filtering, but before the *-top_net_of_hierarchical_group* option is determined. For each net, all global segments which are part of that net will be added to the result collection. Global net segments are all those physically connected across all hierarchical boundaries. Although this option can be used with other options and when there are multiple nets specified, it is best used with a single net. When combined with the *-top_net_of_hierarchical_group* option, you can isolate the highest net segment of a physical net.

-boundary_type *btype*

Specifies what to do when getting nets of boundary pins. This option requires the *-of_objects* option. Allowed values are lower, upper, and both, meaning the net inside the hierarchical block, outside the hierarchical block, or both nets, respectively. The option has no meaning for non-hierarchical pins. Note that The "upper" value is less useful, as getting pins of a hierarchical net will return the pin outside the hierarchical block, and a subsequent get_nets would find the upper hierarchical net. Using upper on a hierarchical pin is the same as omitting the option.

-of_objects *objects*

Creates a collection of nets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell or cell collection. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type net. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **get_nets** command creates a collection of nets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any nets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_nets** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_nets** result to a variable.

When issued from the command prompt, the **get_nets** command behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of **get_nets** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_nets** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the nets that begin with 'NET' in block 'block1'. Although the output looks like a list, it is just a display.

```
pt_shell> get_nets block1/NET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example shows that with a collection of pins, you can query the nets connected to those pins.

```
pt_shell> current_instance block1
block1
pt_shell> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
pt_shell> query_objects [get_nets -of_objects $pinsel]
{"NET1QNX", "NET2QNX"}
```

This example shows how to use the *-top_net_of_hierarchical_group* and *-boundary_type* options. Given the following circuit:

```
u1                                  u2


----------------+              +-------------------
                |              |
 u1/Z -- net5 ---+--- topnet ---+--- net7 -- u1/A
                |              |

----------------+              +-------------------
```

there are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and withing u2 is a pin u1/A being driven by *net7*. These three nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two **get_nets** commands:

```
pt_shell> get_nets * -hierarchical
{"topnet", "u1/net5", "u2/net7"}
pt_shell> get_nets * -hierarchical -top_net_of_hierarchical_group
{"topnet"}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of the *-boundary_type* option. Notice now in this case, the "upper" type does not add value.

```
pt_shell> get_nets -of_objects u2/in
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type upper
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type lower
{"u2/net7"}
pt_shell> get_nets -of_objects u2/in -boundary_type both
{"u2/net7", "topnet"}
pt_shell> get_nets -boundary lower -of_objects \
             [get_pins -of_objects [get_nets topnet]]
{"u1/net5", "u2/net7"}
```

SEE ALSO

```
collections(2)
filter_collection(2)
get_pins(2)
link_design(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_noise_violation_sources

Creates a collection of noise violation sources for custom reporting and other processing. You can assign these pins to a variable and get the desired attribute for further processing.

SYNTAX

```
integer get_noise_violation_sources
   [-above]
   [-below]
   [-low]
   [-high]
   [-nworst_endpoints pin_count]
   [-max_sources_per_endpoint pin_count]
   [-slack_type area | height | area_percent]
   [object_list]
```

Data Types

pin_count	integer
object_list	list

ARGUMENTS

-above

Performs the reporting only above the rails. If this option is combined with the **-low** option, it reports the noise bumps above the low rail. If it is combined with the **-high** option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above the high rail and above the low rail.

-below

Performs the reporting only below the rails. If this option is combined with the**-low** option, it reports the noise bumps below the low rail. If this option is combined with the**-high** option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below the high rail and below the low rail.

-low

Performs the reporting only for the low rail. If this option is combined with the **-above** option, it reports the noise bumps above the low rail. If it is combined with the**-below** option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps for both below and above the low rail.

-high

Performs the reporting only for the high rail. If this option is combined with the **-above** option, it reports the noise bumps above the high rail. If it is combined with the **-below** option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps for both below and above the high rail.

-nworst_endpoints *pin_count*

Specifies the number of endpoint pins to be reported. The default is 1.

-max_sources_per_endpoint *pin_count*

Specifies the number of violation source pins per endpoint to be reported. The default is 1.

-slack_type area | height | area_percent

Specifies the type of slack to be used. The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin.

- **height** (default) - Reports noise slack as the voltage margin.
- **area** - Reports noise slack as the voltage margin multiplied by the noise bump width.
- **area_percent** - Reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

object_list

Specifies the load pins for which the noise reporting is performed. If no pin is specified, reporting is performed on the entire design.

DESCRIPTION

Provides a collection of noise violation source pins for failing noise endpoints in the current design.

EXAMPLES

The following example creates a collection of the worst violation source for the worst endpoint in the current design and assigns it to the viol_pins variable.

```
pt_shell> set viol_pins [get_noise_violation_sources]
```

The following example creates a collection of maximum 10 violation sources for the endpoint ffa/Q above the low rail.

```
pt_shell> get_noise_violation_sources -above -low \
           -max_sources_per_endpoint 10 [get_pin ffa/Q]
```

The following example creates a collection of maximum 10 violation source for the worst two endpoints.

```
pt_shell> get_noise_violation_sources -max_sources_per_endpoint 10 \
           -nworst_endpoints 2
```

SEE ALSO

report_noise(2)

```
report_noise_violation_sources(2)
set_noise_parameters(2)
update_noise(2)
```

get_object_name

Gets the name of objects in the given collection.

SYNTAX

```
string get_object_name
   collection
```

Data Types

collection	string

ARGUMENTS

collection

Specifies the collection.

DESCRIPTION

The **get_object_name** command is a convenient way to get the *full_name* attribute of objects. When multiple objects are passed to the **get_attribute** command, the values are returned as a Tcl list, containing one entry for each object.

EXAMPLES

The following example shows how to use the **get_object_name** command:

```
pt_shell> get_object_name [get_cells i1]
i1
pt_shell> get_attribute [get_cells {i1 i2}] full_name
{i2 i3} i1
pt_shell> get_object_name [get_cells i*]
{i1 i2 i3}
```

SEE ALSO

```
collections(2)
get_attribute(2)
```

get_path_groups

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

SYNTAX

```
collection get_path_groups
   [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns
```

Data Types

expression	string
patterns	list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, it modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, it makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-filter *expression*

Filters the collection with the *expression* value. For any path groups that match the *patterns* values, the expression is evaluated based on the path group's attributes. If the expression evaluates to true, the path group is included in the result.

patterns

Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_path_groups** command creates a collection of path groups from the current design that match certain criteria. The command returns a collection if any path groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Path groups control the optimization cost function and also affect timing reports.

You can use the **get_path_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_path_groups** result to a variable.

When issued from the command prompt, the **get_path_groups** command behaves as though the **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_path_groups** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_path_groups** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following generates a timing report for each path group matching "Z*".

```
pt_shell> foreach_in_collection grp [get_path_groups Z*] { \
?               report_timing -group $grp \
?         }
```

SEE ALSO

```
collections(2)
foreach_in_collection(2)
group_path(2)
query_objects(2)
report_path_group(2)
collection_result_display_limit(3)
```

get_pg_pins

Creates a collection of PG or supply pins.

SYNTAX

```
collection get_pg_pins
   [-hierarchical]
   [-filter expression]
   [-of_objects objects]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   patterns
```

Data Types

```
string expression
list objects
list patterns
```

ARGUMENTS

-hierarchical

Searches for PG or supply pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX find command. For example, if there is a pin block1/inv/vdd, a hierarchical search finds it using inv/vdd. You cannot use the *-hierarchical* option with the *-of_objects* option.

-filter *expression*

Filters the collection with *expression*. For any PG or supply pins that match *patterns* (or *objects*), the expression is evaluated based on the PG or supply pin's attributes. If the expression evaluates to be true, the PG or supply pin is included in the result.

-of_objects *objects*

Creates a collection of PG or supply pins connected to the specified objects. Each object is a leaf cell name or collection, a supply net name or collection, or a pin shape name or collection. The *-of_objects* option and *patterns* are mutually exclusive. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

Makes matches case-insensitive when used with the *-regexp* option. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the asterisks (*) and question marks (?). The *-exact* and *-regexp* options are mutually exclusive.

patterns

Matches PG or supply pin names against patterns. Patterns can include the wildcard characters, such as asterisks (*), question marks (?) or regular expressions, based on the *-regexp* option. Patterns can also include collections of the PG or supply pin. The *patterns* parameter and the *-of_objects* option are mutually exclusive.

DESCRIPTION

The **get_pg_pins** command creates a collection of PG or supply pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any PG or supply pins match the *patterns* or *objects* and passes the filter (if specified). If no objects match the criteria, an empty string is returned.

When used with the *-of_objects* option, the **get_pg_pins** command searches for PG or supply pins connected to any cells or supply nets specified in *objects*.

Regular expression matching is the same as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding the .* symbols to the beginning or end of the expressions as needed.

You can use the **get_pg_pins** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_pg_pins** result to a variable.

When issued from the command prompt, **get_pg_pins** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The implicit query property of **get_pg_pins** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pg_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example shows how to query the VDD pins of cell instances that begin with the character o. Although the output looks like a list, it is just a display.

```
pr_shell> get_pg_pins o*/VDD
{"o_reg1/VDD", "o_reg2/VDD", "o_reg3/VDD", "o_reg4/VDD"}
```

The following example shows that given a collection of cell instances, you can query the PG pins of those cell instances.

```
pr_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pr_shell> query_objects [get_pg_pins -of_objects $csel]
{"o_reg1/VDD", "o_reg1/VSS", "o_reg1/VBP", "o_reg1/VBN"}
```

The following example shows how to get all of the PG pins which are connected to either the VDD or the VDD1 supply net.

```
pr_shell> get_pg_pins -of_object [get_supply_nets {VDD VDD1}]
{"o_reg1/VDD", "o_reg1/VDD", "o_reg2/VDD", "U1/PWR"}
```

The following example shows how to get all of the PG pins which are connected to the inst1/VDD pin shape.

```
pr_shell> get_pg_pins -of_object [get_pin_shapes inst1/VDD]
{"inst1/VDD"}
```

SEE ALSO

collections(2)
filter_collections(2)
get_cells(2)
get_supply_nets(2)
get_pin_shapes(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)

get_pin_shapes

Creates a collection of pin shapes that match the selection criteria.

SYNTAX

```
collection get_pin_shapes
   [-quiet]
   [-exact]
   [-nocase]
   [-filter expression]
   [-within region
     | -touching region
     | -intersect region
     | -at point]
   [patterns | -of_objects objects]
```

Data Types

```
expression    string
region        list
point         list
patterns      list
objects       collection
```

ARGUMENTS

-quiet

Suppresses the reporting of nonterminal (warning and informational) messages.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each pin shape in the collection, the expression is evaluated based on the pin shape's attributes. If the expression evaluates to true, the pin shape is included in the result.

To see the list of pin shape attributes that you can use in the expression, use the **list_attributes -application -class pin_shape** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within ***region***

Creates a collection that contains all pin shapes that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} … {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for pin shapes in the entire design.

-touching ***region***

Creates a collection that contains all pin shapes that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} … {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for pin shapes in the entire design.

-intersect ***region***

Creates a collection that contains all pin shapes that intersect the boundary of the specified region and at least part of the pin shape is outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} … {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for pin shapes in the entire design.

-at ***point***

Creates a collection that contains all pin shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the tool searches for pin shapes in the entire design.

patterns

Creates a collection of pin shapes whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive unless you use the **-nocase** option.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects ***objects***

Creates a collection of pin shapes connected to the specified objects. Each object can be a pg pin.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of pin shapes by selecting pin shapes from the current design that match the specified criteria.

The command returns a collection if any pin shapes match the criteria. If no objects match the criteria, the command returns an empty string.

This command does not support searching across physical hierarchy, this means the blocks in the design are not searched searched beyond the top level.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the all the pin shapes connected to the tc4/sram_target_2/VDD pg pin, as well as the pin shapes on the metal6 layer.

```
prompt> set pg_pin [get_pg_pins tc4/sram_target_2/VDD]
{tc4/sram_target_2/VDD}
prompt> get_pin_shapes -of_objects $pg_pin
{tc4/sram_target_2/VDD}
prompt> get_pin_shapes -filter {layer =~ "M6"} -of_objects $pg_pin
{tc4/sram_target_2/VDD}
```

The following example fetches a pin shape by using its name including its number when multiple pin shapes are associated to the same pin.

```
prompt> get_pin_shapes \
   {"tst_1/sram_target_2/sram_interface_16b/test_so_new 1"}
{tst_1/sram_target_2/sram_interface_16b/test_so_new 1}
```

The following example gets all pin shapes within the specified rectangle.

```
prompt> get_pin_shapes -within {{4 20} {10 30}}
{I_BD_70/gnd I_BD_70/vdd I_BD_70/reset}
```

The following example gets all pin shapes that intersect the specified polygon:

```
prompt>get_pin_shapes \
   -intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{STACK_BLK/MEM_reg_1__0_/VSS STACK_BLK/MEM_reg_9__0_/VDD}
```

SEE ALSO

collections(2)
filter_collection(2)
get_pg_pins(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_pins

Creates a collection of pins from the netlist. You can assign these pins to a variable or pass them into another command.

SYNTAX

```
collection get_pins
   [-hierarchical]
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-leaf]
   [patterns | -of_objects objects]
```

Data Types

```
expression      string
patterns        list
objects         list
```

ARGUMENTS

-hierarchical

Searches for pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the **find** UNIX command. For example, if there is a pin block1/adder/D[0], a hierarchical search finds it using "adder/D[0]". You cannot use the **-hierarchical** option with the **-of_objects** option.

-filter *expression*

Filters the collection with the *expression*. For any pins that match *patterns* (or *objects*), the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

When combined with the **-regexp** option, this makes matches case-insensitive. You can use the **-nocase** option only when you also use the **-regexp** option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The**-exact** and **-regexp** options are mutually exclusive.

-leaf

Returns only leaf (non-hierarchical) pins. If you use the **-leaf** and **-of_objects** options together, and **-of_objects** specifies a collection of nets, the command returns pins on the leaf cells connected to the specified nets; in this case, hierarchical boundaries can be crossed to find these pins.

patterns

Matches pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. The *patterns* option can also include collections of type pin. The *patterns* and **-of_objects** options are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object is a named cell or net, cell collection or pin collection, a named library pin, or a collection of library pins. Note that library pins can not be combined with other types of objects. The**-of_objects** and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the **-hierarchical** option with the **-of_objects** option.

DESCRIPTION

The **get_pins** command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

When used with the **-of_objects** option, the **get_pins** command searches for pins connected to any cells or nets specified in the *objects*. For net objects, there are two variations of pins that will be considered. By default, only pins connected to the net at the same hierarchical level are considered. When combined with the **-leaf** option, only pins connected to the net that are on leaf cells are considered. In this case, hierarchical boundaries are crossed in order to find pins on leaf cells.

If no *patterns* (or *objects*) match any objects, and the current design is not linked, the design automatically links.

When a cell has bus pins, the **get_pins** command can find them in several ways. For example, if cell u1 has a bus "A" with indecies 2 to 0, and the bus_naming_style for your design is "%s[%d]", then to find these pins, you can use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the **regexp** Tcl command. When using the **-regexp** option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_pins** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_pins** command result to a variable.

When issued from the command prompt, the **get_pins** command behaves as though the **query_objects**

command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_pins** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command, use the **get_pins** command as an argument to the **query_objects** command. For example, use this to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the 'CP' pins of cells that begin with 'o'. Although the output looks like a list, it is just a display.

```
pt_shell> get_pins o*/CP
{"o_reg1/CP", "o_reg2/CP", "o_reg3/CP", "o_reg4/CP"}
```

The following example shows that given a collection of cells, you can query the pins connected to those cells.

```
pt_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pt_shell> query_objects [get_pins -of_objects $csel]
{"o_reg1/D", "o_reg1/CP", "o_reg1/CD", "o_reg1/Q", "o_reg1/QN"}
```

The following example shows the difference between getting local pins of a net and leaf pins of net. In this example, NET1 is connected to i2/a and reg1/QN. Cell i2 is hierarchical. Within i2, port a is connected to the U1/A and U2/A.

```
pt_shell> get_pins -of_objects [get_nets NET1]
{"i2/a", "reg1/QN"}
pt_shell> get_pins -leaf -of_objects [get_nets NET1]
{"i2/U1/A", "i2/U2/A", "reg1/QN"}
```

The following example shows how to create a clock using a collection of pins.

```
pt_shell> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

SEE ALSO

```
collections(2)
create_clock(2)
filter_collection(2)
get_cells(2)
link_design(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_ports

Creates a collection of ports from the current design or instance. You can assign these ports to a variable or pass them into another command.

SYNTAX

```
collection get_ports
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [patterns]
   [-of_objects objects]
```

Data Types

```
expression      string
patterns        list
objects         list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* option. For any ports that match the *patterns* option, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

patterns

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type port. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

-of_objects *objects*

Creates a collection of ports connected to the specified objects. Each object is a named net collection. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **get_ports** command creates a collection of ports in the current design or instance that match certain criteria. The command returns a collection if any ports match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If the **port_search_in_current_instance** variable is true, then the **get_ports** command gets the ports of current instance. If the **port_search_in_current_instance** variable is *false* (the default), the **get_ports** command gets the ports of the current design.

Regular expression matching is the same as in the **regexp** Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter the *expression* option; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_ports** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_ports** command result to a variable.

When issued from the command prompt, the **get_ports** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_ports** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command, use the **get_ports** command as an argument to the **query_objects** command. For example, use this to display the object class.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the **all_inputs** and **all_outputs** man pages, which also create collections of ports.

EXAMPLES

The following example queries all input ports beginning with 'mode'. Although the output looks like a list, it is just a display.

```
pt_shell> get_ports "mode*" -filter {direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example sets the driving cell for ports beginning with "in" to an FD2.

```
pt_shell> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern "bidir*".

```
pt_shell> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

SEE ALSO

all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
port_search_in_current_instance(3)
collection_result_display_limit(3)

get_power_domains

Create a collection of power domains in the current design.

SYNTAX

```
string get_power_domains
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [patterns]
   [-of_objects cells]
```

Data Types

```
expression          string
patterns            list
cells               string
```

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any power domains that match *patterns*, the expression is evaluated based on the power domain's attributes. If the expression evaluates to true, the power domain is included in the result. This option works the same as the **filter** command in dc_shell.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

patterns

Matches power domain names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page. The *patterns* and *-of_objects* options are mutually exclusive.

-of_objects *cells*

Returns a collection of power domain that owns the cells. The *patterns* and *-of_objects* options are mutually

exclusive.

DESCRIPTION

The **get_power_domains** command creates a collection of power domains in the current design, that match certain criteria. The command returns a collection handle (identifier) if any power domains match the *patterns* or *-of_objects* value and pass the filter, if specified. If no objects match the criteria, the empty string is returned.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the power domain and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_power_domain SUB_DOMAIN -power_down \
           -power_down_ctrl [get_nets pd_ctrl] \
           -object_list [get_cells mid1]
1
pt_shell> get_power_domain {TOP_DOMAIN SUB_DOMAIN}
{"", ""}

prompt> get_power_domain TOP*
{""}
```

SEE ALSO

report_power_domain(2)

get_power_group_objects

Returns a collection of cells in a power group.

SYNTAX

```
collection get_power_group_objects
   group_names
```

Data Types

group_names	list

ARGUMENTS

group_names

Specifies the power groups of which the collection of cells will be returned.

DESCRIPTION

The **get_power_group_objects** command returns a collection of cells contained by the specified power groups.

EXAMPLES

In the following example, the cells included in the newly created power group are displayed.

```
pt_shell> create_power_group -name my_group [get_cells "u1 u2"]
1
pt_shell> get_power_group_objects my_group
{"u1", "u2"}
```

SEE ALSO

create_power_group(2)

```
remove_power_groups(2)
report_power_groups(2)
```

get_power_switches

Create a collection of UPF power switches in the current design.

SYNTAX

```
string get_power_switches
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-of_objects upf_power_domain|upf_supply_net]
   [patterns]
```

Data Types

```
expression      string
patterns        list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* argument. For any UPF power switches that match the *patterns* argument, the expression is evaluated based on the power switch's attributes. If the expression evaluates to true, the power switch is included in the result. This option works the same as the **filter** command in pt_shell.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

patterns

Matches supply net names against patterns. Patterns can include the wildcard characters "*" (asterisk) and "?" (question mark). For more details about using and escaping wildcards, see the **wildcards** man page. The *patterns* and **-of_objects** arguments are mutually exclusive.

-of_objects

Gets the power switches of the object list. Objects could be combination of upf_power_domain or upf_supply_net. The *patterns* and *-of_objects* arguments are mutually exclusive.

DESCRIPTION

The **get_power_switches** command creates a collection of UPF power switch objects in the current design, that match certain criteria. The command returns a collection handle (identifier) if any supply nets match the *patterns* or **-of_objects** arguments and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates a power switch and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_net SN1 -domain TOP_DOMAIN
1
pt_shell> create_supply_net SN2 -domain TOP_DOMAIN
1
pt_shell> create_power_switch my_switch -domain TOP_DOMAIN \
          -output_supply_port {OUT SN2} \
          -input_supply_port {IN SN2} \
          -on_state {state1 IN {CNTL} } \
          -control_port {ctrl CNTL}

pt_shell> get_power_switches *
{"my_switch"}
```

SEE ALSO

create_power_switch(2)
report_power_switch(2)

get_qtm_ports

Creates a collection of quick timing model (QTM) ports. You can assign these QTM ports to a variable or pass them into another command.

SYNTAX

```
collection get_qtm_ports
   patterns
```

Data Types

patterns	list

ARGUMENTS

patterns

Matches QTM port names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_qtm_ports** command creates a collection of QTM ports from the current QTM model that match certain criteria. The command returns a collection if any QTM ports match the *patterns* option. If no objects match the criteria, an empty string is returned.

To create a QTM port, use the **create_qtm_port** command.

See the **collections** man page for information about collections and the querying of objects.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command applies the **set_qtm_port_load** command to all QTM ports in the model matching "IN*".

```
pt_shell> set_qtm_port_load -value 2.0 [get_qtm_ports "IN*"]
```

The following command creates a delay arc from QTM port A to all QTM ports in the model matching "OUT*".

```
pt_shell> create_qtm_delay_arc -from A -to [get_qtm_ports "OUT*"] -value 3.0
```

SEE ALSO

create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_model(2)
create_qtm_port(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
set_qtm_port_load(2)

get_random_numbers

Generates a list of random numbers for a specified variation.

SYNTAX

```
list get_random_numbers
  -sample_size size_value
  -seed seed_value
  variation_object
```

Data Types

```
size_value          integer
seed_value          integer
variation_object    collection
```

ARGUMENTS

-sample_size *size_value*

Specifies the sample size.

-seed *seed_value*

Specifies the seed for this particular stream. This number must be a positive integer. A different number generates a different sample.

variation_object

Specifies the variation from which to generate random samples. There can only be one object.

DESCRIPTION

This command generates random numbers in a list.

EXAMPLES

In the following example, the returned list contains a sample of size 500 of random numbers generated from a variation with a normal distribution with mean 15.0 and standard deviation 2.0. The seed for this random number

stream is 17777.

```
pt_shell> create_variation -name test -type normal -values { 15.0 2.0 }
_sel21

pt_shell> set rnd_list [get_random_numbers -sample_size 500 \
          -seed 17777 [get_variations test]]
10.613445  14.553537  13.051448  16.382267 ...
```

get_selection

Returns the list of objects currently selected in the GUI or information about the selected objects.

SYNTAX

```
collection get_selection
   [-type object_type]
   [-design design]
   [-more_than more]
   [-fewer_than fewer]
   [-count]
   [-num num]
   [-name slct_bus]
   [-slct_targets target_slct_bus]
   [-slct_targets_operation operation]
   [-create_slct_buses]
   [-type_list]
```

Data Types

object_type	string
design	string
more	integer
fewer	integer
num	integer
slct_bus	string
target_slct_bus	string
operation	string

ARGUMENTS

-type *object_type*

Specifies the type of selected object. If specified, the selection will be filtered with this type. The *object_type* option can be one of the following:

Format	Description
design	design object
port	port object
net	net object
cell	cell object
pin	pin object
path	timing path object

-design *design*

Returns objects that only belong to the *design* option.

-more_than *more*

Returns true if the selection bus contains more than the *more* option.

-fewer_than ***fewer***

Returns true if the selection bus contains more than the *fewer* option.

-count

Returns the number of elements contained in the selection bus.

-num ***num***

Does not return more than the *num* option.

-name ***slct_bus***

Specifies the selection bus from which the selection is taken. The default is "global", which is the default global selection bus.

-slct_targets ***target_slct_bus***

Specifies the name of a selection bus in which to store the result. By default, the result of this command is returned in a collection. If this option is specified, the *target_slct_bus* selection bus is used instead. The *target_slct_bus* option is also returned as a result of the command.

-slct_targets_operation ***operation***

Specifies which operation should be used when storing the result in the *target_slct_bus* selection bus. This is only allowed if you specify the *-slct_targets* option. Legal operations are clear, add, and remove. If the add operation is specified, the result of the **get_selection** command is added to the target selection bus. This is the default behavior. If the clear operation is specified, the target selection bus is first cleared before the result of the **get_selection** command is added. If the remove operation is specified, the result of the **get_selection** command is removed from the target selection bus.

-create_slct_buses

Specifies that a new selection bus is created in which the result is stored. The name of the newly created selection bus is returned.

-type_list

Returns a Tcl list of the types of elements contained in the selection bus.

DESCRIPTION

The **get_selection** command constructs a collection from the objects that are currently selected in the GUI and returns the collection. If no objects are selected, an empty string is returned. If no *-type* option is specified, the command returns a heterogeneous collection of all objects currently selected. If the *-type* option is used, the collection is filtered to a homogeneous collection containing only objects of the type specified.

You can use the **get_selection** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_selection** command result to a variable.

When issued from the command prompt, the **get_selection** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **collection_result_display_limit** variable.

The "implicit query" property of the **get_selection** command provides a fast way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** command (for example, if you want to display the object class), use the **get_selection** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns the collection of the selected cell objects.

```
pt_shell> get_selection -type cell

{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

The next example assigns the collection to the **collect_a** variable, which is passed to another **query_objects** command.

```
pt_shell> set collect_a [get_selection -type cell]

{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}

pt_shell> query_objects $collect_a

{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

The next example assigns the collection to the **pins** variable, which is passed to another **report_timing** command to generate a report on the ten worst paths to the selected pin. The output from the **report_timing** command is not shown here for brevity.

```
pt_shell> set pins [get_selection -type pin]

{"OUT[0]"}

pt_shell> report_timing -to $pins -nworst 10 -maxpaths 10
```

SEE ALSO

collections(2)
filter(2)
filter_collection(2)
query_objects(2)
change_selection(2)

get_si_bottleneck_nets

Identifies crosstalk bottlenecks in the design. This is useful when the major sources of violations come from crosstalk effects.

SYNTAX

```
status get_si_bottleneck_nets
   [-cost_type type]
   [-slack_lesser_than slack_limit]
   [-max_nets number]
   [-significant_digits digits]
   [-nosplit]
   [-include_clock_nets]
   [-minimum_active_aggressors active_aggressor_count]
   [-min]
   [-max]
```

Data Types

type	list
slack_limit	float
number	integer
digits	integer
active_aggressor_count	integer

ARGUMENTS

-cost_type *type*

Sorts the nets based on the cost. The **delta_delay**, **delta_delay_ratio**, **total_victim_delay_bump** are the costs of the victim net. And the **delay_bump_per_aggressor** for aggressor nets of the selected victim nets.

-slack_lesser_than *slack_limit*

Applies the cost function only to the victim nets whose slack is less than the slack limit. The default is 0.0.

-max_nets *number*

Sets the maximum number of nets to be reported. The default is 20.

-significant_digits *digits*

Sets the number of digits to display: Range: 0 to 13. The default is the as same as the **report_default_significant_digits** global variable.

-nosplit

Use to not split lines if column overflows.

-include_clock_nets

Includes the clock nets for bottleneck. By default, clock nets are excluded from the bottleneck analysis.

-minimum_active_aggressors ***active_aggressor_count***

Sets the minimum number of active aggressors for the net to be selected. The default is 1.

-min

Use for the minimum analysis. For example, the *slack_limit* applies to the minimum slack of the net in the net selection.

-max

Use for the maximum analysis. For example, the *slack_limit* applies to the maximum slack of the net in the net selection.

DESCRIPTION

This command has the same behavior as the **report_si_bottleneck** command, but returns nets instead of reporting on them. For more information, see the **report_si_bottleneck** command.

EXAMPLES

The following example gets PrimeTime SI bottleneck. It has an arc with specified start and endpoints.

```
pt_shell> get_si_bottleneck_nets -cost_type delta_delay -significant_digits 6
```

SEE ALSO

```
report_default_significant_digits(3)
report_si_bottleneck(2)
set_coupling_separation(2)
size_cell(2)
```

get_supply_nets

Creates a collection of supply nets in the current design.

SYNTAX

```
collection get_supply_nets
   [-filter expression]
   [-of_objects objects]
   [-quiet]
   [-regexp]
   [-nocase]
   [-expand]
   [patterns]
```

Data Types

```
string expression
list objects
list patterns
```

ARGUMENTS

-filter *expression* Filters the collection with the expression option. For any supply nets that match the patterns options, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result.

-of_objects *objects*

Creates a collection of supply nets connected to the specified objects. Each object is a leaf cell name or collection, or a PG pin name or collection. The -of_objects option and patterns are mutually exclusive.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

-nocase

Makes matches case-insensitive when used with the *-regexp* option. You can use the *-nocase* option only when you also use the *-regexp* option.

-expand

Expands the listed supply nets to include all supply nets that are connected by MTCMOS switch cells. By default the command does not expand the list of supply nets.

patterns

Matches PG or supply pin names against patterns. Patterns can include the wildcard characters, such as asterisks (*), question marks (?) or regular expressions, based on the *-regex* option. Patterns can also include collections of the PG or supply pin. The *patterns* parameter and the *-of_objects* option are mutually exclusive.

DESCRIPTION

The get_supply_nets command creates a collection of supply nets in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply nets match the patterns or -of_objects arguments and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

When used with the -of_objects option, the get_supply_nets command searches for supply nets connected to any domains, cells or PG pins specified in objects.

Regular expression matching is the same as in the Tcl regexp command. When using the -regexp option, take care in the way you quote the patterns and filter expression; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding the .* symbols to the beginning or end of the expressions as needed.

You can use the get_supply_nets command at the command prompt, or you can nest it as an argument to another command (for example, query_objects). In addition, you can assign the get_supply_nets result to a variable.

When issued from the command prompt, get_supply_nets behaves as though query_objects had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable collection_result_display_limit.

For information about collections and the querying of objects, see the collections man page.

EXAMPLES

The following example shows how to query the supply net of cell instances that begin with the character o. Although the output looks like a list, it is just a display.

```
pr_shell> get_supply_nets -of_object [get_cells o*]
{"VSS", "VDD"}
```

The following example shows that given a collection of cell instances, you can query supply nets of those cell instances.

```
pr_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pr_shell> query_objects [get_supply_nets -of_objects $csel]
{"VSS", "VDD"}
```

The following example shows how to get the supply net connected to a cell instance PG pin VDD.

```
pr_shell> get_supply_nets -of_object [get_pg_pins AND12/VDD]
{"VDD"}
```

Assume supply net VDD is connected to virtual supply nets VVDD1 and VVDD2 through MTCMOS switch cells. Supply net VSS is connected to virtual supply nets VVSS1. The following example shows how to get all supply nets connected to VDD and VSS through MTCMOS switch cells.

```
pr_shell> get_supply_nets VDD
{"VDD"}
pr_shell> get_supply_nets VDD -expand
{"VDD", "VVDD1", "VVDD2"}

pr_shell> get_supply_nets VSS
{"VSS"}
pr_shell> get_supply_nets VSS -expand
{"VSS", "VVSS1"}
```

SEE ALSO

```
collections(2)
filter_collections(2)
get_cells(2)
get_pg_pins(2)
query_objects(2)
regexp(2)
collection_result_display_limit(3)
```

get_supply_ports

Create a collection of supply ports in the current design.

SYNTAX

```
string get_supply_ports
   [-filter expression]
   [-of_objects supply_net_list]
   [-quiet]
   [-regexp]
   [-nocase]
   [patterns]
```

Data Types

```
expression      string
supply_net_list list
patterns        list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* option. For any supply ports that match the *patterns* option, the expression is evaluated based on the supply port's attributes. If the expression evaluates to true, the supply port is included in the result. This option works the same as the **filter** command in dc_shell.

-of_objects *supply_net_list*

Creates a collection of supply ports connected to the specified supply nets. The -of_objects option and patterns are mutually exclusive.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* option as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

patterns

Matches supply port names against patterns. Patterns cannot include the wildcard characters "*" (asterisk) and "?" (question mark). For details about using and escaping wildcards, refer to the **wildcards** man page. The

patterns and *-of_objects* options are mutually exclusive.

DESCRIPTION

The **get_supply_ports** command creates a collection of supply ports in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply ports match the *patterns* or *-of_objects* options and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example shows how to get the supply port connected to a supply net VDD.

```
pr_shell> get_supply_ports -of_object [get_supply_nets VDD]
{"VDD"}
```

SEE ALSO

get_supply_net(2)

get_supply_sets

Creates a collection of UPF supply sets in the current design.

SYNTAX

```
collection get_supply_sets
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [patterns]
```

Data Types

```
expression     string
patterns       list
```

ARGUMENTS

-filter *expression*

Filters the collection with the *expression* option. For any UPF supply set that matches the *patterns* option, the expression is evaluated based on the supply set's attributes. If the expression evaluates to true, the supply set is included in the result.

This option works the same as the **filter** command in dc_shell.

-quiet

Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* option as real regular expressions rather than simple wildcard patterns.

-nocase

Performs case-insensitive matching.

patterns

Matches supply set names against patterns. Patterns can include the asterisk (*) and question mark (?) wildcard characters. For information about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

The **get_supply_sets** command creates a collection of UPF supply sets in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply set matches the *patterns* options and pass the filter (if specified). If no object matches the criteria, an empty string is returned.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates a supply set and creates a collection of it.

```
prompt> create_supply_set SS1
1
prompt> get_supply_sets {SS1}
{"SS1"}
prompt> get_supply_sets SS*
{"SS1"}
```

SEE ALSO

create_supply_set(2)

get_switching_activity

Gets switching activity annotation on nets, pins, ports, and cells of the current design.

SYNTAX

```
integer get_switching_activity
   [-state_condition state]
   [-path_sources name_list]
   [-toggle_rate]
   [-glitch_rate]
   [-static_probability]
   [-rise | -fall]
   [-only_related_clock clock_name]
   [-toggle_limit limit]
   [-exclude exclusion_group]
   [-include_only inclusion_group]
   [-average_activity]
   object_list
```

Data Types

state	string
name_list	string
clock_name	string
limit	integer
exclusion_group	string
inclusion_group	string
object_list	list

ARGUMENTS

-state_condition *state*

Specifies the state condition when getting state-dependent toggle rate and glitch rate on pins or state-dependent static probabilities on cells. State dependent toggle rate and glitch rate can be retrieved for a pin when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities for a cell is annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal/leakage power characterization. The state condition should be enclosed between " ". Moreover, if you want to get switching activity information for the default state condition, then the *state* argument should be "default". If the **-state_condition** option is applied to a cell, then only static probability is reported. Only leakage power is associated with cells. If the *state* argument is applied to a pin, then the tool looks for the condition in the internal power tables and report the toggle rate/glitch rate/static probability for that pin.

Given a design object, the command returns the list comprised of object name, the *state* argument, static probability for the *state* argument and the annotation source indicator. If the *state* argument does not represent

a valid state for the cell, then the value of -2 is returned. If the cell has no state-dependent static probability annotation for the *state* argument, then the value of -1 is returned.

-path_sources *name_list*

Specifies the path sources when getting path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path source or sources specified with this argument must be the same as those in the internal power characterization. The **-path** option takes a path condition (a list of input pins). Again, the path condition after the **-path** option should be enclosed between " ". If there are multiple input pins in the path condition, then each input pin should be separated by a space, for example "A B".

Note that if the *name_list* argument is given using the **-path_sources** option, then the *state* argument should also be provided using the **-state_condition** option.

Given a design object, the command returns a list comprised of object name, the **-state_condition** option, source pin name, toggle rate value, toggle rate annotation source indicator, glitch rate value, and glitch rate annotation source indicator. If the combination of the *state* and *name_list* arguments does not represent a valid state-dependent-path-dependent (SDPD) condition for the design object, then the value of -2 is returned. If the design object does not have toggle rate or glitch rate annotation for the given SDPD conditions, then the value of -1 is returned.

-toggle_rate

Specifies the toggle rate (rise and fall, if applicable) which is reported for the given design object. Given a design object, the command returns the list comprised of object name, simple toggle rate value (without glitches), and annotation source indicator. The unit of returned toggle rate value is the number of toggle counts per target technology library time unit. If the object has no toggle rate value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

An annotation source indicator can have the following possible values:

- file: implies that the **-toggle_rate** value was annotated through a SAIF or vcd file
- propagated: implies that the **-toggle_rate** value is propagated using random vectors from annotated design points
- default: implies that the **-toggle_rate** value is default
- UNINITIALIZED: implies that the **-toggle_rate** value is uninitialized
- create_clock: implies that the **-toggle_rate** value was annotated by a **create_clock** command
- set_switching_activity: implies that the **-toggle_rate** value was annotated by a **set_switching_activity** command
- set_case_analysis: implies that the **-toggle_rate** value was annotated by a **set_case_analysis** command
- implied: implies that the activity was implied without random vector propagation from annotated design points

The object name is the full name relative to the current instance.

-glitch_rate

Specifies the glitch rate (rise and fall, if applicable) which is reported for the given design object. Given a design object, the command returns the list comprised of object name, glitch rate value, and an annotation source indicator. The unit of the returned glitch rate value is the number of glitch counts per target technology library times unit. If the object has no glitch rate value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

-static_probability

Specifies the static probability value which is reported for the given design object. Given a design object, the command returns the list comprised of object name, simple static probability and an annotation source indicator. The static probability is defined as the percentage of time the signal is at logic state 1. If the object has no static probability value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

If none of the **-toggle_rate**, **-glitch_rate** and **-static_probability** options are set, then the command assumes all three options to be true, and returns the **-toggle_rate**, **-glitch_rate**, and **-static_probability** values.

-rise

Use with the **-state_condition** option. This option reports either rise or fall transitions. If the **-state_condition** option and *state* argument are given and neither the **-rise** nor **-fall** option is given, then by default the command assumes both to be true. If the **-rise** or **-fall** option is given along with the **-static_probability** option, then the **-rise** and **-fall** options are ignored as static probability does not depend on rise or fall transitions type.

-fall

Use with the **-state_condition** option. This option reports either rise or fall transitions. If the **-state_condition** option and *state* argument are given and neither the **-rise** nor **-fall** option is given, then by default the command assumes both to be true. If the **-rise** or **-fall** option is given along with the **-static_probability** option, then the **-rise** and **-fall** options are ignored as static probability does not depend on rise or fall transitions type.

-only_related_clock *clock_name*

Only nets and objects that are in the clock domain, *clock_name* argument are included. Objects not in the domain are filtered out. When the **-average_activity** option is used, objects are filtered out before averaging takes place.

-toggle_limit *limit*

Changes the definition of what is considered to be low activity. This definition is used by the **-exclude** and **-include_only** options, depending on the criteria given to those options. The value of the argument represents the maximum number of toggles considered to be low activity. The default is 1. The number of toggles on a net is defined as the toggle rate times the simulation duration. For vector free power analysis (where no SAIF file or VCD is used) the default simulation duration is 10000 ns. If a SAIF is used, the duration is taken from the SAIF file. If a VCD file is used to annotate activity, then the duration is the duration of the VCD simulation.

The **-toggle_limit** option has no effect unless the **-exclude** or **-include_only** options are used. The **-toggle_limit** option only affects the behavior of these other options.

-exclude *exclusion_group*

This option filters out objects in the *object_list* argument, so that switching activity is not reported for objects that meet the specified criteria.

When the **-average_activity** option is used, the **-exclude** option serves to filter out nets before the average is taken.

For more information about how to use the **-exclude** and **-include_only** options, read the appropriate section in the man page for the **report_switching_activity** command.

-include_only *inclusion_group*

This option filters out objects in the *object_list* argument, so that switching activity is reported only on objects

that meet the specified criteria.

When the **-average_activity** option is used, the **-include_only** option serves to filter out nets before the average is taken.

For more information about how to use the **-exclude** and **-include_only** options, read the appropriate section in the man page for the **report_switching_activity** command.

-average_activity

When this option is used, the **-static_probability**, **-rise**, **-fall**, and **-path_sources** options cannot be used.

When the **-average_activity** option is used, the *object_list* argument should consist only of hierarchical cells. For each hierarchical cell given, the average toggle rate and glitch rate of nets within the hierarchical cell is computed and reported.

In this case, the **-include_only** and **-exclude** options play the role of filtering out nets in the hierarchical cell before the average toggle rate is computed.

Static probabilities are never averaged. If the **-average_activity** option is used, only average toggle rate and average glitch rate is reported. Since the nets in the hierarchical cell might have different sources for the activity information, the activity source for the toggle and glitch rate reports as "averaged".

object_list

Specifies a list of nets, pins, ports, or cells in the current design for which the **-static_probability**, **-toggle_rate**, and **-glitch_rate** options switching activity values are reported.

When the **-average_activity** option is used, the *object_list* argument should consist of hierarchical cells whose average switching activity on nets in the cell is computed.

DESCRIPTION

This command is used to report different kinds of switching activity information on design nets, ports, pins, and cells. These include simple toggle rate, glitch rate, and static probability on nets, ports, and pins; state and path dependent toggle rate and glitch rate on cell pins, and state dependent static probabilities on cells.

Toggle rates, glitch rates, and static probabilities are reported using the **-toggle_rate**, **-glitch_rate** and **-static_probability** options, respectively. The toggle rate, glitch rate, and static probability can be made state dependent by specifying the state condition with the **-state_condition** option. The toggle rate and glitch rate can be made path dependent by specifying the path source or sources with the **-path_sources** option.

The design objects that are annotated with switching activity can only be specified explicitly as a list of objects. Note that the *object_list* argument should always be provided to the command.

Average activity statistics for a hierarchical block, or portions of a hierarchical block is obtained using the **-average_activity** option.

For more statistics on the switching activity annotation on the current design, use the **report_switching_activity** command.

If the **power_analysis_mode** variable is set to **averaged** or **leakage_variation**, and if a VCD file is read with the **read_vcd** command (without the **-pipe** option), then the tool annotates the activity from the VCD before reporting the switching activity. In addition, unannotated objects where the activity can be derived accurately without random vector simulation (for example, clock nets, Qbar pins where the Q pin is annotated, constant nets) is reported by the **report_switching_activity** command as having the **implied** argument activity. Activity

propagation for other unannotated nodes does not happen until the **update_power** command runs.

If the **power_analysis_mode** variable is set to **time_based**, then the **get_switching_activity** command does not have access to toggle rate information before running the **update_power** command. The command reports whether a net is uninitialized or whether it annotates from the VCD file. But since the VCD file has not been read yet, the toggle rate (and other values) is reported as -1.

Use of the **-exclude** and **-include_only** filtering options allow you to filter out objects before reporting switching activity. This is useful, for instance, to report only nets with no switching activity. In this case, the *object_list* argument could be all nets in the design (obtained by the **[get_net * -hierarchical -top]** argument), while the **-include_only {no_switching_activity}** option filters the large set to return a list of only those nets with no switching activity.

EXAMPLES

The following **get_switching_activity** command reports simple toggle rate value, glitch rate value, and static probability value on all design input ports.

```
pt_shell> get_switching_activity -toggle_rate -glitch_rate \
         -static_probability [all_inputs]
pt_shell> get_switching_activity [all_inputs]
```

The following example reports different switching activity values on design output ports.

```
pt_shell> get_switching_activity -toggle_rate \
          [all_outputs]

pt_shell> get_switching_activity -glitch_rate \
          [all_outputs]

pt_shell> get_switching_activity -static_probability \
          [all_outputs]

pt_shell> get_switching_activity -toggle_rate \
          -static_probability [all_outputs]

pt_shell> get_switching_activity -glitch_rate \
          -static_probability [all_outputs]

pt_shell> get_switching_activity -glitch_rate \
          -toggle_rate [all_outputs]

pt_shell> get_switching_activity -glitch_rate \
          -toggle_rate -static_probability [all_outputs]
```

The following example reports state dependent static probabilities on the cell or1:

```
pt_shell> get_switching_activity -static_probability \
          -state_condition "A & B" [get_cell or1]

pt_shell> get_switching_activity -state_condition "A & B" \
          [get_cell or1]

pt_shell> get_switching_activity -static_probability \
          -state_condition "A & ! B" [get_cell or1]

pt_shell> get_switching_activity -static_probability \
          -state_condition "! A & B" [get_cell or1]
```

```
pt_shell> get_switching_activity -static_probability \
          -state_condition "! A & ! B" [get_cell or1]

pt_shell> get_switching_activity -static_probability \
          -state_condition "default" [get_cell or1]
```

The following example reports simple and path dependent toggle rates and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> get_switching_activity -toggle_rate -glitch_rate \
          [get_pin xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \
          -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> get_switching_activity -rise -fall -toggle_rate \
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> get_switching_activity -rise -toggle_rate \
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> get_switching_activity -fall -toggle_rate \
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \
          -path_sources "B" -state_condition "A" [get_pin xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \
          -path_sources "A B" -state_condition "default" [get_pin xor1/Y]
```

The following example reports the average switching activity on certain nets within the hierarchical cell "state_machine1". The nets used are those that are driven by sequential cells and have annotated switching activity.

```
pt_shell> get_switching_activity -toggle_rate -average_activity \
          -include_only {sequential & annotated} [get_cell state_machine1]
```

The following example reports the switching activity on every pin that is an output of a sequential cell. This might be useful for checking to see that switching activity annotation to sequential cells happened properly when you invoke a command, such as the **read_saif** command.

```
pt_shell> get_switching_activity  \
          -include_only {sequential} [get_pin * -hierarchical]
```

The following example reports the switching activity on every pin that is an output of a sequential cell or a black box and does not have annotated activity. This could be useful for generating a list of pins that you might want to annotate before propagation of switching activity.

```
pt_shell> get_switching_activity  \
          -include_only {sequential,black_box & !annotated} [get_pin * -hierarchical]
```

The following example reports the switching activity on every net that has fewer than 10 toggles during simulation. Note that the **get_switching_activity** command reports toggle rates, not toggle counts. To obtain the toggle count of a net, multiply the rate by the simulation time.

```
pt_shell> get_switching_activity  -toggle_limit 10 \
          -include_only {low_activity} [get_net * -hierarchical -top]
```

The following example reports on a net in the **averaged** power analysis mode. The net is in the VCD file, and the **read_vcd** command had previously been given. This example runs the **get_switching_activity** command before and after the **update_power** argument. The example shows that before power analysis, on the first invocation of an activity reporting command, the VCD activity is applied, then the toggle rate and source are reported. After power analysis, the toggle rate is also reported.

```
pt_shell> get_switching_activity "z[31]"
Information: Reading file vcd.dump.gz to annotate toggle rates on the design...

=====================================================================
Summary:
Total number of nets = 1629
Number of annotated nets = 1629 (100.00%)
Total number of leaf cells = 1339
Number of fully annotated leaf cells = 1339 (100.00%)
=====================================================================

 {"z[31]" 0.00330065 file 0 file 0.525616 file}
pt_shell> get_switching_activity "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.525616 file}
pt_shell> update_power
Information: Running average power analysis...
1
pt_shell> get_switching_activity "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.525616 file}
```

The following example reports on a net in the **time_based** power analysis mode. The net is in the VCD file. This example runs the **get_switching_activity** command before and after the **update_power** argument. The example shows that before power analysis, the activity source is known (file), but the toggle rate is unknown and is reported as -1. After the **update_power** argument, the toggle rate is known and is reported.

```
pt_shell> get_switching_activity  "z[31]"
 {"z[31]" -1 file -1 file -1 file}
pt_shell> update_power
Information: The waveform options are:
               File name:      primetime_px.fsdb
               File format:    fsdb
               Time interval:  0.01ns
               Hierarchical level:     1

Information: Power analysis is running, please wait ...

Information: analysis is done for time window (0ns - 9998.03ns)

1
pt_shell> get_switching_activity  "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.526482 file}
```

SEE ALSO

```
read_saif(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_switching_activity(2)
```

get_taps

Creates a collection of tap objects.

SYNTAX

```
collection get_taps
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [patterns| -of_objects objects]
   [-result result_name]
   [-within query_region]
```

Data Types

```
string expression
list objects
list patterns
string result_name
list query_region
```

ARGUMENTS

-filter *expression*

Filters the collection with the value specified by expression. For any taps that match the patterns (or objects) argument, the expression is evaluated based on the specified tap attributes. See the **rail_attributes** man page for a detailed list of tap attributes. If the expression is evaluated to be true, the tap is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the value of the pattern arguments as real regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. Note that the -regexp and -exact options are mutually exclusive.

-nocase

Makes the pattern matching case-insensitive. Use the -nocase option only when you also use the -regexp option.

-exact

Disables the simple pattern matching. This is used when searching for objects that contain wildcard characters, like asterisks (*) and question marks (?) symbols. The -exact and -regexp options are mutually exclusive.

patterns

Matches tap names against patterns. Patterns can include wildcard characters asterisks (*) and question marks (?) symbols or regular expressions, depending on the use of the -regexp option. Patterns can also include collections of tap types. Note that the patterns argument and the -of_objects option are mutually exclusive.

-of_objects *objects*

Creates a collection of taps that are attached to the specified objects. The *objects* can be cell names or collections, supply net names or collections, net shapes, pin_shapes or terminals. Note that the -of_objects option and patterns are mutually exclusive.

-result *result_name*

Specifies which result context is to be used to process names or patterns in the pattern parameter. When not specified, taps are collected from the current session. The *result_name* can either be the current rail or integrity result. If the -result option is specified without a pattern for the tap parameter, the command collects all taps from the specified *result_name*. If the -result option is specified with a pattern for the tap parameter, the tap names in the pattern parameter are looked up and collected from within the specified *result_name*. The -result option cannot be used with the -of_object option .

-within *query_region*

Creates a collection of taps that lie within the specified query region consisting of a bounding box and a layer. The *query_region* is a list of the following format: {left bottom right top layer}. *left*, *bottom*, *right*, and *top* are in user distance unit. *layer* can be either layer number or name.

DESCRIPTION

The **get_taps** command creates a collection of taps that are currently in the session context or result context of the current design. See the **tap_contexts** man page for a detailed description about tap contexts. The command returns a collection if any taps match the patterns or objects and pass the filter (if specified). If no object matches the criteria, an empty collection is returned.

When used with the the -of_objects option, the **get_taps** command searches for taps that are connected to any cells, supply nets, net shapes, pin shapes or terminals as specified with the -of_objects option.

Using the -regexp option to enable regular expression matching is the same as using the TCL regexp command. When using the -regexp option, be careful of the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression matching begins at the beginning of an object name and stops at the end of an object name. You can widen the search simply by adding the period and asterisk symbols (.*) to the beginning or end of the expressions as needed.

You can use the **get_taps** command at the command prompt, or nest it as an argument to another command, like the **query_objects** command. In addition, you can assign the result of the **get_taps** run to a variable.

When issued from the command prompt, the **get_taps** command behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. To change the setting, use the *collection_result_display_limit* variable. The "implicit query" property of the **get_taps** command provides a fast and simple way to display cells in a collection. However, if you want the flexibility provided by **query_objects** (for example, if you want to display the object class), use the **get_taps** command as an argument to the **query_objects** command.

For information about collections and querying objects, see the **collections** and **tap_context** man pages.

The *supply_net* attribute of a tap object returned by the **get_taps** command is a collection of objects of type *upf_supply_net*. This type is identical to the type of object returned by the **get_supply_nets** command. Note the prefix *upf_* here does not imply it is related to the UPF standard.

EXAMPLES

The following example queries all taps that match a pattern TAP?2, except for the TAP_42.

```
pr_shell> get_taps -filter tap_name!=TAP_42 TAP_?2
{"TAP_12", "TAP_22", "TAP_32", "TAP_52"}
```

The following example queries the taps that are connected to a collection of the cells.

```
pr_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pr_shell> query_objects [get_taps -of_objects $csel]
{"TAP_32", "TAP_33", "TAP44"}
```

The following example gets all of the taps that are connected to the VDD or VDD1 supply nets.

```
pr_shell> get_taps -of_object [get_supply_nets {VDD VDD1}]
{"TAP_1", "TAP_2", "TAP_3"}
```

The following example gets all of the taps that touch the PATH#12544 net shape.

```
pr_shell> get_taps -of_object [get_net_shapes PATH#12544]
{"TAP_1", "TAP_2"}
```

The following example gets taps from the rail result context after running the **calculate_rail_voltage** command.

```
pr_shell> calculate_rail_voltage VDD
pr_shell> get_taps -result rail
{"tap_a", "tap_b"}
pr_shell> get_taps *a -result rail
{"tap_a"}
```

The following example gets taps from the integrity result context after running the **check_supply_net_integrity** command.

```
pr_shell> check_supply_net_integrity VDD
pr_shell> get_taps -result integrity
{"integrity_tap1", "tap_a", "tap_b"}
pr_shell> get_taps integrity* -result integrity
{"integrity_tap1"}
```

The following example gets all of the taps that are connected to the VDD supply net using a filter expression.

```
pr_shell> get_taps -filter supply_net.full_name==VDD
{"TAP_1"}
```

The following example gets all of the taps that lie within the specified bounding box on M1 layer.

```
pr_shell> get_taps -within {0.0 0.0 450.0 30.0 M1}
{"TAP_4" "TAP_6"}
```

SEE ALSO

```
collections(2)
collection_result_display_limit(3)
create_taps(2)
current_integrity_result(2)
current_rail_result(2)
filter_collections(2)
get_cells(2)
get_net_shapes(2)
get_pin_shapes(2)
get_supply_nets(2)
get_terminals(2)
query_objects(2)
rail_attributes(2)
regexp(2)
remove_taps(2)
report_integrity_result(2)
report_rail_result(2)
report_taps(2)
tap_contexts(2)
write_taps(2)
```

get_terminals

Creates a collection of terminals that match the specified criteria.

SYNTAX

```
collection get_terminals
   [-quiet]
   [-exact]
   .prod all
   [-nocase]
   [-filter expression]
   .prod icc
   [-within region
     | -touching region
     | -intersect region
     | -at point]
   [patterns
     | -of_objects supply_port_list
     | -object_id object_id]
   .prod all
```

Data Types

```
expression     string
region      list of points
point      point
patterns       list
supply_port_list      collection
object_id      integer
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

.prod all

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each terminal in the collection, the expression is

evaluated based on the terminal's attributes. If the expression evaluates to true, the terminal is included in the result.

To see the list of terminal attributes that you can use in the expression, use the **list_attributes -application -class terminal** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all terminals that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for terminals in the entire design.

-touching *region*

Creates a collection that contains all terminals that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for terminals in the entire design.

-intersect *region*

Creates a collection that contains all terminals that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for terminals in the entire design.

-at *point*

Creates a collection that contains all terminals at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches for terminals in the entire design.

patterns

Creates a collection of terminals whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

.prod icc

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern. .prod syn The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern. .prod all

-of_objects *supply_port_list*

Creates a collection of terminals connected to the specified supply ports.

.prod icc

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern. .prod syn The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern. .prod all

-object_id *object_id*

Creates a collection that contains the terminal with the specified Milkyway object ID. To determine the Milkyway object ID, use the **get_attribute** command to return the value of the **object_id** attribute.

The *patterns*, **-of_objects**, and **-object_id** arguments are mutually exclusive; you can specify only one. If you do not specify either of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of terminals by selecting terminals from the current design that match the specified criteria.

The command returns a collection if any terminals match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a collection of terminals named reset or with a prefix of clock:

```
prompt> get_terminals [list reset clock*]
{reset clock}
```

.prod icc

The following examples creates a collection that contains the terminal with the Milkyway object ID of 4104.

```
prompt> get_terminals -object_id 4104
{reset}
```

The following example returns the terminals within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}:

```
prompt> get_terminals -within {{2 2} {25 25}}
{reset}
```

The following example gets terminals that intersect the specified polygon:

```
prompt> get_terminals \
   -intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{CCEN}
```

.prod all

SEE ALSO

collections(2)
filter_collection(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_timing_arcs

Creates a collection of timing arcs for custom reporting and other processing. You can assign these timing arcs to a variable and get the desired attribute for further processing.

SYNTAX

```
string get_timing_arcs
   [-to to_list]
   [-from from_list]
   [-of_objects object_list]
   [-filter expression]
   [-quiet]
```

Data Types

```
to_list          list
from_list        list
object_list      list
expression       string
```

ARGUMENTS

-to *to_list*

Specifies a list of to pins or to ports to get timing arcs for. All backward arcs from the specified pins or ports are considered.

-from *from_list*

Specifies a list of from pins or from ports to get timing arcs for. All forward arcs from the specified pins or ports are considered.

-of_objects *object_list*

Specifies cells or nets to get timing arcs for. If a cell is specified, all cell_arcs of that cell are considered. If a net is specified, all net_arcs of that net are considered.

-filter *expression*

Specifies a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value by means of an operator.

-quiet

Specifies that all messages are to be suppressed.

DESCRIPTION

The **get_timing_arcs** command creates a collection of arcs for custom reporting or other operations. Use the **foreach_in_collection** command to iterate among the arcs in the collection. You can use the **get_attribute** command to obtain information about the arcs. You can also use the filter expression to filter the arcs that satisfy the specified conditions. The following attributes are supported on timing arcs:

```
delay_max_fall
delay_max_rise
delay_min_fall
delay_min_rise
from_pin
is_annotated_fall_max
is_annotated_fall_min
is_annotated_rise_max
is_annotated_rise_min
is_cellarc
is_disabled
is_user_disabled
mode
object_class
sdf_cond
sense
to_pin
```

One attribute of a timing arc is from_pin which is the pin or port from which the timing arc begins. In the same way, the to_pin is the pin or port at which the timing arc ends. In order to get more information about the from_pin and the to_pin, use the **get_attributes** command. The *object_class* attribute holds the name of the timing arc class *timing_arc*.

EXAMPLES

The following procedure gets the same arguments as the **get_timing_arcs** command and prints out some of the attributes of the timing arcs.

```
proc format_float {number {format_str "%.2f"}} {

  switch -exact -- $number {
    UNINIT { }
    INFINITY { }
    default {set number [format $format_str $number]}
  }
  return $number;
}


proc show_arcs {args} {

  set arcs [eval [concat get_timing_arcs $args]]
  echo [format "%15s    %-15s  %8s %8s  %s" "from_pin" "to_pin" \
            "rise" "fall" "is_cellarc"]
  echo [format "%15s    %-15s  %8s %8s  %s" "--------" "------" \
            "----" "----" "----------"]

  foreach_in_collection arc $arcs {
    set is_cellarc [get_attribute $arc is_cellarc]
```

```
    set fpin [get_attribute $arc from_pin]
    set tpin [get_attribute $arc to_pin]
    set rise [get_attribute $arc delay_max_rise]
    set fall [get_attribute $arc delay_max_fall]

    set from_pin_name [get_attribute $fpin full_name]

    set to_pin_name [get_attribute $tpin full_name]
    echo [format "%15s -> %-15s  %8s %8s  %s" \
        $from_pin_name $to_pin_name \
        [format_float $rise] [format_float $fall] \
        $is_cellarc]
  }
}


pt_shell> show_arcs -of_objects ffa

       from_pin    to_pin               rise     fall  is_cellarc
       --------    ------               ----     ----  ----------
         ffa/CP -> ffa/D                0.85     0.85  true
         ffa/CP -> ffa/D                0.40     0.40  true
         ffa/CP -> ffa/Q                1.34     1.42  true
         ffa/CP -> ffa/QN               2.38     1.98  true
         ffa/CD -> ffa/Q                1.00     0.82  true
         ffa/CD -> ffa/QN               1.78     1.00  true

pt_shell> show_arcs -from ffa/CP

       from_pin    to_pin               rise     fall  is_cellarc
       --------    ------               ----     ----  ----------
         ffa/CP -> ffa/D                0.85     0.85  true
         ffa/CP -> ffa/D                0.40     0.40  true
         ffa/CP -> ffa/Q                1.34     1.42  true
         ffa/CP -> ffa/QN               2.38     1.98  true

pt_shell> show_arcs -from ffa/Q

       from_pin    to_pin               rise     fall  is_cellarc
       --------    ------               ----     ----  ----------
          ffa/Q -> QA/A                 0.00     0.00  false

pt_shell> show_arcs -to ffa/*

       from_pin    to_pin               rise     fall  is_cellarc
       --------    ------               ----     ----  ----------
         ffa/CP -> ffa/D                0.85     0.85  true
         ffa/CP -> ffa/D                0.40     0.40  true
            a/Z -> ffa/D                0.00     0.00  false
            CLK -> ffa/CP               0.00     0.00  false
          RESET -> ffa/CD               0.00     0.00  false
         ffa/CP -> ffa/Q                1.34     1.42  true
         ffa/CD -> ffa/Q                1.00     0.82  true
         ffa/CP -> ffa/QN               2.38     1.98  true
         ffa/CD -> ffa/QN               1.78     1.00  true
```

SEE ALSO

collections(2)
foreach_in_collection(2)

```
get_attribute(2)
```

get_timing_paths

Creates a collection of timing paths for custom reporting or other processing.

SYNTAX

```
collection get_timing_paths
   [-from from_list
     | -rise_from rise_from_list
     | -fall_from fall_from_list]
   [-to to_list
     | -rise_to rise_to_list
     | -fall_to fall_to_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-exclude exclude_list
     | -rise_exclude rise_exclude_list
     | -fall_exclude fall_exclude_list]
   [-delay_type delay_type]
   [-nworst paths_per_endpoint]
   [-max_paths max_path_count]
   [-group path_group_list]
   [-unique_pins]
   [-slack_greater_than minimum_slack]
   [-slack_lesser_than maximum_slack]
   [-ignore_register_feedback feedback_slack_cutoff]
   [-include_hierarchical_pins]
   [-trace_latch_borrow]
   [-trace_latch_forward]
   [-pba_mode none | path | exhaustive]
   [-normalized_slack]
   [-start_end_pair]
   [-cover_design]
   [-dont_merge_duplicates]
   [-pre_commands pre_command_string]
   [-post_commands post_command_string]
   [-path_type format]
   [-attributes attribute_list]
   [-tag_paths_filtered_by_pba tag_name]
   path_collection
```

Data Types

```
from_list                 list
rise_from_list            list
fall_from_list            list
to_list                   list
rise_to_list              list
fall_to_list              list
through_list              list
rise_through_list         list
fall_through_list         list
exclude_list              list
```

```
rise_exclude_list         list
fall_exclude_list         list
delay_type                string
paths_per_endpoint        integer
max_path_count            integer
path_group_list           list
minimum_slack             float
maximum_slack             float
feedback_slack_cutoff     float
pre_command_string        string
post_command_string       string
format                    string
attribute_list            list
tag_name                  string
path_collection           collection
```

ARGUMENTS

-from ***from_list***

Collects only paths that start from the specified list of pins, ports, nets, or cell instances; or from startpoints clocked by the named clocks. Valid startpoints are clock pins of sequential devices and input pins of the design.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must start with a rising transition. If you specify a clock object, the option selects startpoints clocked by the named clock and launched by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from ***fall_from_list***

Same as the **-rise_from** option, except that the path must start with a falling transition.

-to ***to_list***

Collects only paths that end at the specified list of pins, ports, nets, or cell instances; or at endpoints clocked by the named clocks. Valid endpoints are data input pins of sequential devices and output pins of the design.

Due to the infrastructure of the timing analyzer, the **-to** option runs more efficiently than the **-from** option. As a result, endpoint-oriented path gathering (based on using the **-to** option) outperforms equivalent startpoint-oriented path gathering (based on using the **-from** option).

-rise_to ***rise_to_list***

Same as the **-to** option, except that the path data must end with a rising transition. If you specify a clock object, the option selects endpoints clocked by the named clock and captured by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_to ***fall_to_list***

Same as the **-rise_to** option, except that the path data must end with a falling transition.

-through ***through_list***

Collects only paths in which the data goes through the named pins, ports, cell instances, or nets. You can use this option multiple times in a command.

When a single **-through** list contains multiple objects, the data path can go through any one of the listed

objects. When you use multiple **-through** options, the data path must meet each of the **-through** options in sequence.

For example, the following command gets a path that starts at A1, then passes through either B1 or B2, then passes through C1, and ends at D1.

```
pt_shell> report_timing -from A1 -through {B1 B2} -through C1 -to D1
```

If advanced analysis through transparent latches is enabled (**timing_enable_through_paths** set to **true**), and the pins specified in the last through switch are latch loop breakers, then the timing paths collected are constrained by the worst of the closing edges at the latch or by a required value computed using logic downstream of the latch loop breaker.

-rise_through *rise_through_list*

Same as the **-through** option, except that it applies only to paths with a rising transition at the specified objects.

-fall_through *fall_through_list*

Same as the **-through** option, except that it applies only to paths with a falling transition at the specified objects.

-exclude *exclude_list*

Excludes all paths in which the data goes from, through, or to the specified list of pins, ports, nets, or cell instances; or excludes all paths belonging to the specified list of path tag sets. It checks only data paths (not clock paths) for excluded objects.

Using the **-exclude** option can result in significantly longer runtimes. To reduce the runtime impact, use this option together with other topological restriction options such as **-from** and **-through**.

The **-exclude** option does not check borrowing paths considered by the **-trace_latch_borrow** option.

-rise_exclude *rise_exclude_list*

Same as the **-exclude** option, but applies only to rising transitions at the named pins, ports, nets, or cell instances.

-fall_exclude *fall_exclude_list*

Same as the **-exclude** option, but applies only to rising transitions at the named objects.

You can use no more than one of the **-exclude**, **-rise_exclude**, or **-fall_exclude** options in a command.

-delay_type *delay_type*

Specifies the type of path delay constraint to consider for finding and sorting paths with the worst slack:

- **max** (default) -- max delay (setup constraint)
- **min** -- min delay (hold constraint)
- **min_max** -- both min and max delay (stored min first, max second)
- **max_rise** -- max delay, rising at data path endpoint
- **max_fall** -- max delay, falling at data path endpoint
- **min_rise** -- min delay, rising at data path endpoint
- **min_fall** -- min delay, falling at data path endpoint

-nworst ***paths_per_endpoint***

Collects up to the specified number of worst paths per endpoint. The default is 1. A larger value results in a larger collection and more runtime. Allowed values are 1 to 2000000.

If you set **-nworst** to any number greater than 1, the command automatically does the following:

- Implicitly sets **-max_paths** equal to the **-nworst** setting if the **-max_paths** option is not used in the command, so that at least one set of multiple paths to an endpoint can be reported
- Implicitly sets **-slack_lesser_than 0.0** if the **-slack_lesser_than** and **-slack_greater_than** options are not used in the command, so that only violating paths are reported

-max_paths ***max_path_count***

Collects up to the specified maximum total number of paths among all path groups. The default is equal to the **-nworst** setting, or 1 if the **-nworst** option is not used. Allowed values are 1 to 2000000.

If you set **-max_paths** to any number greater than 1, the command implicitly sets **-slack_lesser_than 0.0** if the **-slack_lesser_than** and **-slack_greater_than** options are not used in the command, so that only violating paths are reported.

The **-nworst** option controls the maximum number of paths collected *per endpoint*, whereas the **-max_paths** option controls the *overall total maximum number* of paths collected. The **-max_paths** setting should be at least as large as the **-nworst** setting, so by default the tool implicitly sets **-max_paths** to the same value as **-nworst** if you do not use the **-max_paths** option in the command.

-group ***path_group_list***

Collects only paths that belong to the specified path groups. Path groups are created by the **create_clock** and **group_path** commands. Without the **-group** option, the **get_timing_paths** command collects paths without considering path groups. For example, the **get_timing_paths** command alone creates a collection containing the single worst path in the design, whereas **get_timing_paths -group [get_path_groups]** creates a collection containing multiple paths, consisting of the single worst path in each path group.

-unique_pins

Collects only the single worst timing path through any given sequence of pins. No other paths are collected for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not collected for that sequence of pins.

For non-unate logic such as XOR gates, using this option greatly reduces the number of paths collected because of the large number of possible rising/falling edge combinations through each sequence of pins.

Using this option can require longer runtimes when used with the **-nworst** option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is collected and counted toward the total number of requested paths.

-slack_greater_than ***minimum_slack***

Collects only paths with slack greater than the specified *minimum_slack* value; these paths have a negative slack better than the specified *minimum_slack* value (or a positive slack that is farther from causing a violation). This option is intended to be used with the **-slack_lesser_than** option to collect paths within a specific range of slack.

-slack_lesser_than ***maximum_slack***

Collects only paths with slack less than the specified *maximum_slack* value; these paths have a negative slack worse than the specified *maximum_slack* value (or a positive slack that is closer to causing a violation).

By default, the **report_timing** command collects the single path with the worst slack; or if the **-nworst** or **-max_paths** option is set to a value greater than 1, the command considers all paths with negative slack, as if **-slack_lesser_than** is set to 0.0.

The **-slack_lesser_than** and **-slack_greater_than** options act as filters to restrict the paths that are collected, so the number of paths collected can be fewer than the number specified with the **-nworst** and **-max_paths** options. If no paths meet the slack criteria, no paths are collected.

-ignore_register_feedback ***feedback_slack_cutoff***

Ignores noninverting timing loops that start and end at the same register pin that holds a value. To be ignored, the data-to-output arc and the output-to-data path must be either both inverting or both noninverting. This option applies to minimum delay as well as maximum delay constraints. Paths are ignored only if they have a slack less than the specified *feedback_slack_cutoff* value.

This option acts as a filter to restrict the paths that are collected, so the number of paths collected can be fewer than the number specified with the **-nworst** and **-max_paths** options.

-include_hierarchical_pins

Causes the collected paths to include points for the hierarchical pins crossed, as well as all leaf pins in the path. The hierarchical pins are included only for documentation purposes; the **report_timing** command reports them as having zero incremental delay and they do not affect the timing results.

-trace_latch_borrow

Controls the type of data collected for a path that starts at a transparent latch. If the path startpoint borrows from the previous path segment, using this option causes the path data to include the set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. Each path segment is maintained separately, including the time borrowed and lent and the endpoints of the path segment.

Without the **-trace_latch_borrow** option, the command does not include upstream borrowing latches when in default latch analysis mode or borrowing latch loop breakers when in advanced latch analysis mode (**timing_enable_through_paths** variable set to true) as path segments in the timing report.

-trace_latch_forward

Controls the type of data collected for a path that ends at a transparent latch D pin. If advanced analysis through transparent latches is enabled (**timing_enable_through_paths** is set to true) and this transparent latch D pin is constrained by a downstream required time, this option causes the data to include the paths in the fanout of this D pin that led to the downstream required constraint. These paths can include multiple path segments whereby the constraint is the result of propagating through more than one latch constrained by the downstream required time. In this case, each path segment is maintained separately, showing the downstream required time for that path segment.

Without the **-trace_latch_forward** option, the command does not trace the path beyond the specified endpoint.

-pba_mode none | path | exhaustive

Specifies one of the following path-based timing analysis modes:

- **none** (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- **path** - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths. To control the ordering of the paths (either by original graph-based slack or recalculated path-based slack), set the **pba_path_mode_sort_by_gba_slack** variable.

- **exhaustive** - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode. You cannot use the **exhaustive** mode together with the **-start_end_pair**, **-cover_design**, or *path_collection* options.

In ordinary graph-based (default) timing analysis, the tool considers both the worst arrival time and worst slew among all signals feeding into a path, even when the worst arrival and worst slew come from different signals.

In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

-normalized_slack

Collects and sorts paths using normalized slack instead of absolute slack. Normalized slack is the absolute slack divided by an idealized maximum allowable propagation delay (typically the clock period). To use this option, the timing update must be performed while the **timing_enable_normalized_slack** variable is set to **true**.

-start_end_pair

Collects the worst path for each startpoint-endpoint pair in the design, including all violating startpoint-endpoint pairs by default, or all startpoint-endpoint pairs with slack worse than the value specified by the **-slack_lesser_than** option if that option is used. The number of paths collected is limited to 2000000.

The **-start_end_pair** option overrides the default behavior of collecting only the single worst path in the design. Instead, the command collects the worst timing path of each violating startpoint-endpoint pair, which can result in a very large collection and long runtimes. (Note that the Design Compiler and IC Compiler tools differ from this behavior by respecting the default limits on the number of paths collected and by allowing usage of the **-nworst** and **-max_paths** with **-start_end_pair**.)

To limit the collection size and runtime to reasonable amounts, use the **-slack_lesser_than** option and specify a slack limit that is only slightly greater than the worst slack in the design. You can also restrict the scope of the analysis by using the **-from**, **-to**, and similar options. You cannot use the **-start_end_pair** option with the **-nworst**, **-max_paths**, **-unique_pins**, **-slack_greater_than**, or **-ignore_register_feedback** options.

Using the **-start_end_pair** option requires any **-from**, **-rise_from**, or **-fall_from** option to specify only valid timing startpoints, and any **-to**, **-rise_to**, or **-fall_to** option to specify only valid timing endpoints; wildcard characters that specify invalid startpoints or endpoints are not allowed. In other words, the **timing_report_always_use_valid_start_end_points** variable is treated as set to **true**, irrespective of its actual setting.

The **-start_end_pair** option sorts the paths first by endpoint and then by slack, with the paths sharing the same endpoint stored together in order of increasing slack.

-cover_design

Collects the worst path through each violating pin in the design, so that the collected paths cover every violating pin in the design. A pin is deemed to be violating if its slack is less than zero, or less than the value specified with the **-slack_lesser_than** option if that option is used. The slack of a pin is the worst slack among all paths that start at, pass through, or end on that pin. The maximum number of paths collected is 2000000.

You can perform path-based analysis on the generated collection to get path-based accuracy that covers all known violating pins in the design, without the runtime cost of performing a full exhaustive path-based analysis.

The following options cannot be used with **-cover_design**:

-nworst

```
-max_paths
-unique_pins,
-ignore_register_feedback
-start_end_pair
-pba_mode exhaustive
path_collection
```

-dont_merge_duplicates

Prevents the merging of duplicate paths across multiple scenarios when the PrimeTime tool is invoked with the **-multi_scenario** option.

By default, when the same path is collected in more than one scenario, the tool keeps only the single most critical instance of that path in the merged collection.

If you use this option, the tool does not merge duplicate instances of the same path, but instead collects the most critical instances of the path from all scenarios. Because the number of paths collected might limited by **-nworst**, **-max_paths**, or other command options, the resulting merged collection might be more focused on a portion of the design that is critical in multiple scenarios, and less evenly spread out across different portions of design.

-pre_commands *pre_command_string*

Specifies a list of commands, separated by semicolons, to be executed in the worker (slave) context before execution of the **get_timing_paths** command. This option is available only if you invoke the PrimeTime tool with the **-multi_scenario** option. The maximum size of a command is 1000 characters.

-post_commands *post_command_string*

This option is like the **-pre_commands** option, except that the commands are executed after (instead of before) the **get_timing_paths** command.

-path_type *format*

Specifies the types of data points stored for each path in the path collection. You can query the data points as path attributes or report them with the **report_timing** command.

By default, the command stores the full data path, including all intermediate points and their incremental and cumulative delay, together with the launch and capture clock arrival times, data required time, and slack. To also store the clock path points leading up the launch and capture points, including points leading up to generated clocks, use the **-path_type full_clock_expanded** option.

-attributes *attribute_list*

Specifies the attributes to be returned to the master from a slave collection. By default, a slave collection returns only the full_name, scenario_name, and object_class attributes. To control the amount of data retrieved from the slave, use this option together with the **set_distributed_parameters -collection_levels** command.

The **-attributes** option is available only if you invoke the PrimeTime tool with the **-multi_scenario** option.

-tag_paths_filtered_by_pba *tag_name*

When used with the **-pba_mode path** option, the **-tag_paths_filtered_by_pba** option causes tagging of all paths originally found by path-based analysis and later discarded by path-based analysis, using the specified tag name. In future **get_timing_paths** and **report_timing** commands, you can exclude the tagged paths from further analysis by specifying the tag name as an argument to the **-exclude** option.

path_collection

Performs path-based analysis on the specified paths and returns a new path collection. You can use this option

only with the **-pba_mode path** option. This option is mutually exclusive with options that control the selection of paths to collect.

DESCRIPTION

The **get_timing_paths** command creates a collection of paths for custom reporting or other operations. Most of the **get_timing_paths** command options are shared by **report_timing**, and the behavior of these shared options is identical. The order in which paths are returned from **get_timing_paths** matches the reported timing path order of **report_timing**.

You can pass the generated collection directly to another command, or you can set a variable to the collection for later reporting. For example,

```
pt_shell> sizeof_collection [get_timing_paths -nworst 4 -max_paths 20]
Warning: report_timing has satisfied the max_paths criteria. ...
20

pt_shell> set mypaths [get_timing_paths -nworst 4 -max_paths 20]
Warning: report_timing has satisfied the max_paths criteria. ...
Information: Defining new variable 'mypaths'. (CMD-041)
 ...
pt_shell> sizeof_collection $mypaths
20
pt_shell> get_attribute $mypaths slack
-1.157432 -0.987188 -0.984205 -0.971914 -0.968931 -0.873521 ...
pt_shell> report_timing $mypaths
(20 path timing reports displayed)
```

To iterate over the paths in the collection, use the **foreach_in_collection** command. To obtain information about paths, use the **get_attribute**, **report_attribute**, and collection commands. To get a list of timing path attributes, use the following command:

```
pt_shell> list_attributes -application -class timing_path
 ...
Attribute Name              Object        Type       Properties  Constraints
-------------------------------------------------------------------------------
arrival                     timing_path   float      A
capture_clock_paths         timing_path   collection A
clock_uncertainty           timing_path   float      A
close_edge_adjustment       timing_path   float      A
common_path_pessimism       timing_path   float      A
...
```

One attribute of a timing path is the **points** collection. A point corresponds to a pin or port along the path. Iterate through these points with the **foreach_in_collection** command and collect attributes on them by using the **get_attribute** command. To get a list of timing path attributes, use the following command:

```
pt_shell> list_attributes -application -class timing_point
 ...
Attribute Name               Object         Type     Properties  Constraints
-------------------------------------------------------------------------------
annotated_delay_delta        timing_point   float    A
annotated_delta_transition   timing_point   float    A
aocvm_coefficient            timing_point   float    A
applied_derate               timing_point   float    A
arrival                      timing_point   float    A
 ...
```

For information about creating collections and iterating over the elements of a collection, see the examples in the

EXAMPLES section and the man pages of the **collections** and **foreach_in_collection** commands.

To report the dominant and overridden timing exceptions related to the timing path, use the **report_timing** command with the **-exceptions** option. To get more information about the exceptions that apply to the path, you can create a path collection with the **get_timing_paths** command and query the timing path attributes:

```
Attribute                         Possible Reason
-----------------------------------------------------------------------
dominant_exception                false_path / min_max_delay /
                                  multicycle_path
startpoint_unconstrained_reason   no_launch_clock / dangling_start_point /
                                  fanout_of_disabled
endpoint_unconstrained_reason     no_capture_clock / dangling_end_point /
                                  fanin_of_disabled
```

EXAMPLES

You can find some detailed examples of custom timing reports in:

```
$SYNOPSYS_ROOT/auxx/pt/examples/tcl
```

where $SYNOPSYS_ROOT is the installation directory for PrimeTime.

The following procedure reports the startpoint name, endpoint name, and the slack of the worst path in each path group.

```
proc custom_report_worst_path_per_group {} {
  echo [format "%-20s %-20s %7s" "From" "To" "Slack"]
  echo "-------------------------------------------------------"
  foreach_in_collection path [get_timing_paths] {
    set slack [get_attribute $path slack]
    set startpoint [get_attribute $path startpoint]
    set endpoint [get_attribute $path endpoint]
    echo [format "%-20s %-20s %s" [get_attribute $startpoint full_name] \
      [get_attribute $endpoint full_name] $slack]
  }
}
```

```
pt_shell> custom_report_worst_path_per_group

>From                 To                     Slack
-------------------------------------------------------
ffa/CP               QA                   0.1977
ffb/CP               ffd/D                3.8834
```

The following example shows Total Negative Slack, Total Positive Slack, and Worst Negative Slack for the current design.

```
proc report_design_slack_information {} {
  set design_tns 0
  set design_wns 100000
  set design_tps 0
  foreach_in_collection group [get_path_groups *] {
    set group_tns 0
    set group_wns 100000
    set group_tps 0
    foreach_in_collection path [get_timing_paths -nworst 10000 -group $group] {
      set slack [get_attribute $path slack]
      if {$slack < $group_wns} {
```

```
          set group_wns $slack
          if {$slack < $design_wns} {
             set design_wns $slack
          }
        }
        if {$slack < 0.0} {
          set group_tns [expr $group_tns + $slack]
        } else {
          set group_tps [expr $group_tps + $slack]
        }
      }
      set design_tns [expr $design_tns + $group_tns]
      set design_tps [expr $design_tps + $group_tps]
      set group_name [get_attribute $group full_name]
      echo [format "Group '%s' Worst Negative Slack : %g" $group_name $group_wns]
      echo [format "Group '%s' Total Negative Slack : %g" $group_name $group_tns]
      echo [format "Group '%s' Total Positive Slack : %g" $group_name $group_tps]
      echo ""
    }
    echo "-------------------------------------------"
    echo [format "Design Worst Negative Slack : %g" $design_wns]
    echo [format "Design Total Negative Slack : %g" $design_tns]
    echo [format "Design Total Positive Slack : %g" $design_tps]
  }

pt_shell> report_design_slack_information

Group 'CLK' Worst Negative Slack : -3.1166
Group 'CLK' Total Negative Slack : -232.986
Group 'CLK' Total Positive Slack : 4.5656

Group 'vclk' Worst Negative Slack : -4.0213
Group 'vclk' Total Negative Slack : -46.1982
Group 'vclk' Total Positive Slack : 0

-------------------------------------------
Design Worst Negative Slack : -4.0213
Design Total Negative Slack : -279.184
Design Total Positive Slack : 4.5656
```

If you use the **-pba_mode** option, path recalculation (path-based analysis) is used during the path search, and the worst recalculated paths meeting your criteria are returned. This option requires that a path search to be performed to ensure that the paths being returned truly are the worst paths. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases. Large **-nworst** values can significantly increase the time needed for the search.

To see the worst path in the design with path-based analysis applied:

```
pt_shell> report_timing [get_timing_paths -pba_mode exhaustive]
```

To report all endpoints in the design which fail after considering path-based analysis:

```
pt_shell> set mypaths [get_timing_paths -pba_mode exhaustive \
            -slack_lesser_than 0 -max_paths 1000]
pt_shell> report_timing $mypaths
```

SEE ALSO

```
collections(2)
create_clock(2)
foreach_in_collection(2)
get_attribute(2)
group_path(2)
read_sdf(2)
report_timing(2)
set_case_analysis(2)
set_operating_conditions(2)
timing_report_always_use_valid_start_end_points(3)
```

get_user_shapes

Creates a collection of user shapes that match the specified criteria.

SYNTAX

```
collection get_user_shapes
   [-quiet]
   [-regexp | -exact]
   [-nocase]
   [-within region
     | -touching region
     | -intersect region]
   [-filter expression]
   [patterns]
```

Data Types

```
region        list of points
expression    string
patterns      list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each user shape in the collection, the expression is evaluated based on the user shape's attributes. If the expression evaluates to true, the user shape is included in the result.

To see the list of shape attributes that you can use in the expression, use the **list_attributes -application -class shape** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all user shapes that are completely inside the specified region and do not overlap the boundary. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-touching *region*

Creates a collection that contains all user shapes that are inside the specified region, including those that overlap the boundary. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-intersect *region*

Creates a collection that contains all user shapes that intersect the boundary of the specified region and at least part of the cell is outside of the specified region. The region boundary can be a rectangle or a polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, and **-intersect** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

This option is mutually exclusive with the options **-within**, **-touching** and **-intersect**.

patterns

Creates a collection of user shapes whose names match the specified patterns. User shapes are named based on the type of shape they are:

- Rectangle shapes are named RECTANGLE#*n*
- Trapezoid shapes are named TRAPEZOID#*n*
- Path shapes are named PATH#*n*
- Polygon shapes are named POLYGON#*n*
- Wire shapes are named WIRE#*n*

The *n* value in the shape names is an integer value.

Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive unless you use the **-nocase** option.

If you do not specify this argument, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of user shapes by selecting user shapes from the current design that match the specified criteria.

This command does not support searching across physical hierarchy, this means the blocks in the design are not searched beyond the top level.

The command returns a collection if any user shapes match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the shapes in a specified area. Although the output looks like a list, it is only a display.

```
prompt> get_user_shapes -within {{10.5 22.3} {32.0 40}}
{RECTANGLE#100 TRAPEZOID#120 POLYGON#145}
```

The following example gets the user shapes that intersect with the specified polygon:

```
prompt> get_user_shapes \
   -intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{POLYGON#9936}
```

The following example queries the shapes that have the specified route type.

```
prompt> get_user_shapes \
   -filter {route_type=="P/G Std. Cell Pin Conn"}
{PATH#1134336 PATH#1134337 PATH#1134338 PATH#1134339 PATH#1134340}
```

The following examples get the shapes inside the block referred by u1.

```
prompt> get_user_shapes u1/PATH#12345
{u1/PATH#12345}

prompt> get_user_shapes u1/HWIRE#*
{u1/HWIRE#12301 u1/HWIRE#12302 u1/HWIRE#12303}
```

SEE ALSO

collections(2)
filter_collection(2)
get_net_shapes(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

get_vias

Creates a collection by selecting vias from the current design that match the specified criteria.

SYNTAX

```
collection get_vias
   [-quiet]
   [-filter expression]
   [-within region
     | -intersect region
     | -touching region
     | -at point]
   [patterns
     | -of_objects supply_nets]
```

Data Types

```
expression    string
region        list
point         string
patterns      list
supply_nets   collection
```

ARGUMENTS

-quiet

Suppresses warning and error messages.

-filter *expression*

Filters the collection with the specified expression. For each via in the collection, the expression is evaluated based on the via's attributes. If the expression evaluates to true, the via is included in the result.

To see the list of via attributes that you can use in the expression, use the **list_attributes -application -class via** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-within *region*

Creates a collection that contains all vias that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you

do not specify any of these options, the command searches the entire design.

-intersect ***region***

Creates a collection that contains all vias that intersect the boundary of the specified region and at least part of the via is outside of the specified region. The region boundary can be a rectangle or a polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-touching ***region***

Creates a collection that contains all vias that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon. For details about specifying the region boundary, see the DESCRIPTION section.

The coordinate unit is specified in the technology file; typically the unit is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

-at ***point***

Creates a collection that contains all vias at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The **-within**, **-touching**, **-intersect**, and **-at** options are mutually exclusive; you can specify only one. If you do not specify any of these options, the command searches the entire design.

patterns

Creates a collection of vias whose names match the specified patterns. Vias created by the tool use the following naming convention: VIA#*n*, VIA_ARRAY#*n*, where *n* is an integer. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case-sensitive.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects ***supply_nets***

Creates a collection of vias connected to the specified supply nets.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

DESCRIPTION

This command creates a collection of vias by selecting vias from the current design that match the specified criteria.

When using the **-within**, **-touching**, or **-intersect** option, you can specify the region boundary as a rectangle or polygon. The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-

left and upper-right corners of the rectangle. The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

This command does not support searching across physical hierarchy, this means the blocks in the design are not searched beyond the top level.

The command returns a collection if any vias match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates collections of vias that are connected to the n300 net:

```
prompt> get_vias -of_objects [get_supply_nets VDD]
{VIA#6766}
```

The following example gets all vias within specified rectangle:

```
prompt> get_vias -within {{100 100} {105 105}}
{VIA#12345}
```

The following example gets all vias that intersect the specified polygon:

```
prompt>get_vias \
   -intersect {{30 20} {50 20} {50 30} {40 30} {40 40} {30 40} {30 20}}
{VIA#9936 VIA#1234}
```

SEE ALSO

get_supply_nets(2)
collections(2)
filter_collection(2)
get_attribute(2)
list_attributes(2)
query_objects(2)
collection_result_display_limit(3)
wildcards(3)

getenv

Returns the value of a system environment variable.

SYNTAX

```
string getenv
   variable_name
```

Data Types

variable_name	string

ARGUMENTS

variable_name

Specifies the name of the environment variable to be retrieved.

DESCRIPTION

The **getenv** command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **getenv**, **setenv**, and **printenv** environment commands are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using the **setenv** command, you see the new value within the application and within any new child processes you initiate from the application using the **exec** command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

See the **set**, **unset**, and **printvar** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **getenv** returns you to your home directory:

```
prompt> set home [getenv "HOME"]
/users/disk1/bill

prompt> cd $home

prompt> pwd
/users/disk1/bill
```

In the following example, **setenv** changes the value of an environment variable:

```
prompt> getenv PRINTER
laser1

prompt> setenv PRINTER "laser3"
laser3

prompt> getenv PRINTER
laser3
```

In the following example, the requested environment variable is not defined. The error message shows that the Tcl variable **env** was indexed with the value UNDEFINED, which resulted in an error. In the second command, **catch** is used to suppress the message.

```
prompt> getenv "UNDEFINED"
Error: can't read "env(UNDEFINED)": no such element in array
        Use error_info for more info. (CMD-013)

prompt> if {[catch {getenv "UNDEFINED"} msg]} {
         setenv UNDEFINED 1
       }
```

SEE ALSO

catch(2)
exec(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
unsetenv(2)
unset(2)

group_path

Groups paths for cost function calculations and reporting.

SYNTAX

```
status group_path
   -name group_name
   -default
   [-weight weight_value]
   [-from from_list]
   [-rise_from rise_from_list]
   [-fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list]
   [-rise_to rise_to_list]
   [-fall_to fall_to_list]
   [-comment comment_string]
```

Data Types

group_name	string
weight_value	float
from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list
comment_string	string

ARGUMENTS

-name *group_name*

Specifies a name for the group. If a group with this name already exists, PrimeTime adds the paths or endpoints to that group. If a group with this name does not exist, PrimeTime creates a new group. You must specify the **-name** option unless you use the **-default** option; the **-name** and **-default** options are mutually exclusive.

-default

Specifies that endpoints or paths are moved to the default group and removed from the current group. You must specify the **-default** option unless you use the **-name** option; the **-default** and **-name** options are mutually exclusive.

-weight ***weight_value***

Specifies a cost function weight for this group. The *weight_value* value must be a number between 0.0 and 100.0, the default is 1.0. A weight of 0.0 eliminates the paths in this group from cost function calculations. Do not use very small values (for example, 0.0001). Smaller values can prevent PrimeTime from implementing small improvements to the design. If you specify the **-weight** option when you add members to an existing group, the new weight for the group is used.

-from ***from_list***

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to ***fall_to_list***

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-through ***through_list***

Specifies a list of path throughpoints (ports, pins, cells, or nets).

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

-rise_through ***rise_through_list***

It is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects.

-fall_through ***fall_through_list***

It is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects.

-comment ***comment_string***

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

Groups a set of paths or endpoints for cost function calculations in optimization and analysis. Design Compiler uses the cost function to direct optimization. Path groups also affect the output of the **report_timing** and **report_constraint** commands.

The delay cost function is the sum of all groups (weight * violation), where violation is the cost of the worst path in the path group. If no violation occurs within a group, the group cost is zero. Groups enable you to specify a set of paths to optimize even though there might be larger violations in another group. When you specify endpoints, all paths leading to those endpoints are grouped. PrimeTime creates default groups for clock gating checks and asynchronous preset/clear checks named **"**clock_gating_default**"** and **"**async_default**"** respectively. Paths belonging to these groups may moved to user-defined groups through the **group_path** command.

If you specify a clock in the **object_list** variable, all endpoints related to that clock are included in the group. The **create_clock** command automatically creates a group for a new clock with a weight of 1.0 and the same name as the clock name.

The weight of the default group is 1.0. If you do not specify the *weight_value* value for a new group, the default is 1.0.

To undo the **group_path** command, use the **remove_path_group** command. To report path group information for a design, use the **report_path_group** command.

EXAMPLES

The following example groups all endpoints clocked by CLK1A or CLK1B into a new group 'group1' with weight = 2.0.

```
pt_shell> group_path -name group1 -weight 2.0 -to {CLK1A CLK1B}
```

The following example adds OUT1 and ff34/D to the existing path group named 'ADDR'.

```
pt_shell> group_path -name ADDR -to {OUT1 ff34/D}
```

The following example removes OUT1 and CLK2 from existing groups and places them in the default group.

```
pt_shell> group_path -default -to {OUT1 CLK2}
```

The following example groups all paths from inputs I1 and I2 to outputs O5 and O7 into a group named 'serious' with weight 10.0.

```
pt_shell> group_path -name serious -weight 10.0 -from {I1 I2} -to {O5 O7}
```

SEE ALSO

```
create_clock(2)
current_design(2)
remove_path_group(2)
report_constraint(2)
report_path_group(2)
reset_design(2)
set_input_delay(2)
set_output_delay(2)
```

gui_start

Starts the PrimeRail GUI.

SYNTAX

```
string gui_start
   [-file name_of_script_file]
   [-no_windows]
```

Data Types

```
string name_of_script_file
```

ARGUMENTS

-file *name_of_script_file*

The given script file is sourced before the GUI starts.

-no_windows

The GUI starts without showing the default window.

DESCRIPTION

This command starts the PrimeRail GUI from the pr_shell prompt. The command is ignored if the PrimeRail GUI has already been started.

EXAMPLES

The following example starts the PrimeRail GUI.

```
pr_shell> gui_start
```

SEE ALSO

gui_stop(2)

gui_stop

Stops the PrimeRail GUI.

SYNTAX

`string` **`gui_stop`**

ARGUMENTS

None.

DESCRIPTION

This command stops the PrimeRail GUI and returns to the pr_shell prompt. The command is ignored within scripts.

EXAMPLES

The following example stops the PrimeRail GUI and returns to the pr_shell prompt.

```
pr_shell> gui_stop
```

SEE ALSO

`gui_start(2)`

help

Displays quick help for one or more commands.

SYNTAX

```
string help
   [-verbose]
   [-groups]
   [pattern]
```

Data Types

```
pattern      string
```

ARGUMENTS

-verbose

Displays the command options; for example *command_name* **-help**.

-groups

Displays a list of command groups only.

pattern

Displays commands matching the specified pattern.

DESCRIPTION

The **help** command is used to get quick help for one or more commands or procedures. This is not the same as the **man** command that displays reference manual pages for a command. There are many levels of help.

By typing the **help** command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to **help**. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type **help** followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the **-verbose** option to get syntax help for one or more commands. Use the **-groups** option to show only the command groups in the application. This option cannot be combined with any other option.

EXAMPLES

The following example lists the commands by command group:

```
prompt> help
Specify a command name or wild card pattern to get help on individual
commands. Use the '-verbose' option to get detailed option help for a
command. Commands also provide help when the '-help' option is passed to
the command.'

You can also specify a command group to get the commands available in
that group. Available groups are:

  Procedures:                Miscellaneous procedures
  Help:                      Help commands
  Builtins:                  Generic Tcl commands
 ...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> help procedures
 ls                 # List files
 sh                 # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with *a*:

```
prompt> help a*
 alias                # Create a command which expands to words.
 append               # Builtin
 array                # Builtin
```

This example displays option information for the **source** command:

```
prompt> help -verbose source
 source               # Read a file and execute it as a script
   [-echo]                (Echo all commands)
   [-verbose]             (Display intermediate results)
   file_name              (Script file to read)
```

SEE ALSO

man(2)
sh_help_shows_group_overview(3)

history

Displays or modifies the commands recorded in the history list.

SYNTAX

```
string history
   [-h]
   [-r]
   [argument_list]
```

Data Types

```
argument_list      list
```

ARGUMENTS

-h

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the **source** command. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

-r

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

argument_list

Additional arguments to **history** (see DESCRIPTION).

DESCRIPTION

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." The most commonly used forms of the command are described below. You can combine each with either the **-h** or **-r** option, but not both.

- With no arguments, the **history** command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.
- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this

option is a nonstandard extension to Tcl.

- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

  history keep *count*

Tcl supports many additional forms of the **history** command. See the "Advanced Tcl History" section below.

EXAMPLES

The following examples show the basic forms of the **history** command. The first is an example of how to limit the number of events shown using a single numeric argument:

```
prompt> history 3
     7  set base_name "my_file"
     8  set fname [format "%s.db" $base_name]
     9  history 3
```

Using the **-r** option creates the history listing in reverse order:

```
prompt> history -r 3
     9  history -r 3
     8  set fname [format "%s.db" $base_name]
     7  set base_name "my_file"
```

Using the **-h** option removes the leading numbers from each history line:

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

Advanced Tcl History

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." When specifying an event to the **history** command, the following forms may be used:

- A number, which if positive, refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event; for example, **-1** refers to the previous event, **-2** to the one before that, and so on. Event **0** refers to the current event.
- A string selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the **string match** command.

The **history** command can take any of the following forms:

history

Same as **history info**, described below.

history add *command [exec]*

Adds the *command* argument to the history list as a new event. If **exec** is specified (or abbreviated), the command is also executed and its result is returned. If **exec** is not specified, an empty string is returned.

history change *newValue [event_number]*

Replaces the value recorded for an event with *newValue*. The *event_number* specifies the event to replace, and defaults to the *current* event (not event **-1**). This command is intended for use in commands that implement new forms of history substitution and want to replace the current event (that invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear

Erases the history list. The current keep limit is retained. The history event numbers are reset.

history event *[event_number]*

Returns the value of the event given by *event_number*. The default value of *event_number* is **-1**.

history info *[count]*

Returns a formatted string, giving the event number and contents for each of the events in the history list except the current event. If *count* is specified, then only the most recent *count* events are returned.

history keep *[count]*

Changes the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid

Returns the number of the next event to be recorded in the history list. Use this for printing the event number in command-line prompts.

history redo *[event_number]*

Reruns the command indicated by *event* and returns its result. The default value of *event_number* is **-1**. This command results in history revision. See the following section for details.

History Revision

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the **substitute** and **words** history operations have been removed. The **clear** operation was added.

The **redo** history option results in much simpler "history revision." When this option is invoked, the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying the history, use the **event** operation to retrieve an event, and use the **add** operation to add it to history and execute it.

identify_interface_logic

Sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic.

SYNTAX

```
status identify_interface_logic
   [-ignore_ports port_list]
   [-auto_ignore]
   [-latch_level levels]
   [-context_borrow]
   [-keep_ignored_fanout]
   [-include_pins pin_list]
   [-critical_pins]
   [-include_all_net_pins]
   [-traverse_disabled_arcs]
```

Data Types

```
port_list      list
levels         integer
pin_list       list
```

ARGUMENTS

-ignore_ports *port_list*

Specifies a list of input or output ports whose fanout or fanin is to be ignored when placing the *is_interface_logic_pin* attribute. Substitute the list of ports you want for the *port_list* value. Clock ports are automatically ignored; you do not have to specify them in this list.

You can use this option to exclude the fanout of ports connected to chip-level nets (for example, scan enable and reset) from affecting the contents of interface logic. If these nets are not explicitly ignored, they cause unnecessary logic (for example, internal registers) to be part of the interface logic on the current design. You can also use this option to selectively generate the interface logic for a subset of the ports on a block; for example, an interface logic model (ILM) can model only the timing behavior on a subset of the ports on a block.

By default all nonclock input and all output ports are used in identifying the contents of interface logic. The **-ignore_ports** and **-auto_ignore** options are mutually exclusive.

-auto_ignore

Enables automatic determination of ports whose fanout should be ignored when placing the *is_interface_logic_pin* attribute. A port is ignored if the percentage of the total registers in the design in the transitive fanout of the port exceeds a specified threshold percentage contained in the **ilm_ignore_percentage** variable. The default is 25.

You can use this option to help you identify the test enable and reset ports of your design. Before using it,

examine the current value of the **ilm_ignore_percentage** variable and reset it if necessary. Note that the **-auto_ignore** option might potentially ignore ports you do not want to ignore, or fail to ignore ports you want to ignore. Carefully read the messages issued by this command when you use this option to see which ports have been ignored and what percentage of registers to which they fanned out.

The **-ignore_ports** and **-auto_ignore** options are mutually exclusive.

-latch_level *levels*

Specifies the number of logic levels over which time borrowing can occur for latch chains that are a part of the interface logic. Substitute the number of levels you want for *levels*. By default, all latches found in interface logic are assumed to be potential borrowers. Thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic. Note that this is the opposite of the default behavior of the **extract_model** command.

When you use this option, note that the value of *level* must account for the borrowing behavior of latches only on interface timing paths. If *n* represents a latch level, then *n* + 1 represents the number of latches maintained in latch chains that originate at input ports. The *n* + *1*th latch functions as an edge-triggered register and decouples I/O paths from internal paths. Use this option only when there are latches in the interface logic for a block. Specifying this option might potentially result in less accurate models, because not all latches are allowed to borrow. The **-context_borrow** and **-latch_level** options are mutually exclusive.

-context_borrow

Specifies that latch borrowing at the interface should be established based on the current context defined for the design. Therefore, latches that borrow based on arrival times defined on input ports and clocks defined on the design are traced through, but path tracing stops at nonborrowing latches. The **-context_borrow** and **-latch_level** options are mutually exclusive.

-keep_ignored_fanout

Specifies that the fanout from ignored input ports to interface logic is to be maintained in the model. Thus, ignored ports are not used to identify interface logic, but connections between ignored ports and interface logic are preserved. Using this option results in larger models, particularly in conjunction with the **-include_all_net_pins** option of the **write_ilm_netlist** command. Timing slacks for ignored ports might differ from those reported on the original netlist because connections from ignored ports to internal registers are not preserved in the model.

-include_pins *pin_list*

Specifies a list of pins that must be included in the interface logic. Substitute the list of pins you want for *pin_list*. This option can be used to optionally add internal pins to the interface logic. After the interface logic is determined, this list of pins is added to the interface logic. You can use this option to define additional interface logic pins that will otherwise be removed in the model. There will not be any affect on pins that are already in the interface logic. Use this option in conjunction with the **all_fanin** and **all_fanout** commands to include a particular cone of logic in the model.

-critical_pins

Specifies that critical pins interface logic must be identified. This option allows you to extract a model that keeps only the pins in the critical paths of the design.

You can use this option to extract a critical pins interface logic. The model generated contains the interface logic pins in the critical paths of the design. The model is compact compared to normal interface logic models, but might not be as accurate outside of the current context. This model can be used only to do critical path analysis. It might take much longer to generate this model than the normal model.

-include_all_net_pins

Specifies that all pins on nonclock interface logic nets and propagated clock interface logic nets are to be

included in the netlist. Use this option if the interface logic model (ILM) is used in a non-SDF flows and delay calculations are performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net.

The **-include_all_net_pins** option does not affect nonpropagated clock nets; that is, nets in the clock network that have user-specified source latency. This option is similar to the same option of the **write_ilm_netlist** command except that here it adds the extra pins to the interface logic. Therefore, if you query the interface logic pins by using the **get_ilm_objects** command, you notice these extra pins have been added.

The **write_ilm_netlist** command writes these pins only to the Verilog netlist, but does not add them to the interface logic. It is recommended that you use this option while extracting a critical pins interface logic model.

-traverse_disabled_arcs

Specifies that the interface logic should not be affected by disabled arcs/pins on the design. If specified, this option forces the traversal of disabled arcs while determining interface logic.

DESCRIPTION

Sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic. This is the first step in the generation of an ILM.

The interface logic on a block contains all cells whose timing is affected by, or affects, the external environment of a block. The following list describes such parts of the interface logic:

- All cells in timing paths that lead from input ports to registers or output ports that terminate the paths.
- All cells in timing paths that lead to output ports from registers or input ports that originate the paths.
- Clock trees that drive interface registers; including any registers in the clock tree. Clock-gating circuitry is part of interface logic if it is driven by external ports, but not if it is driven by registered outputs on a block.

Notice that interface logic does not include internal register-to-register paths and logic on a block associated only with these paths.

This command implicitly performs an update_timing on the design if required.

You can review the objects you have identified as interface logic by using the **get_ilm_objects** command. When you are satisfied with the interface logic designations, you can nondestructively generate a flattened Verilog netlist for the interface logic model (ILM) by using the **write_ilm_netlist** command. To generate script and back-annotation files for the ILM, use the **write_ilm_script**, **write_ilm_parasitics**, and **write_ilm_sdf** commands.

You can reset the *is_interface_logic_pin* attribute by executing the **identify_interface_logic** command again. You can remove all user-defined attributes, including the *is_interface_logic* attribute by using the **reset_design** command.

EXAMPLES

The following example places the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins in the fanin/fanout of ports *port1*, *port2*, and *port3*.

```
pt_shell> identify_interface_logic \
```

```
-ignore [get_ports {port1 port2 port3}]
```

The following example additionally includes three levels of logic from internal pin *ff/Q*.

```
pt_shell> set ilm_pins [all_fanout -levels 3 \
-flat [get_pin ff/Q]]
pt_shell> identify_interface_logic \
-include_pins $ilm_pins
```

The following example extracts a critical pins interface logic model.

```
pt_shell> identify_interface_logic -critical_pins \
-include_all_net_pins
```

The following example places the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins identified by the **-auto_ignore** option. Because the value of the **ilm_ignore_percentage** variable is currently 35, pins are ignored if the percentage of the total design registers in their transitive fanout is greater than 35. The **-latch_level** option with a value of *1* states that the first latch encountered in I/O paths might potentially borrow, but the second latch can be treated as an edge-triggered device. Thus, two levels of latches are maintained in the interface logic.

```
pt_shell> printvar ilm_ignore_percentage
ilm_ignore_percentage = "35"
pt_shell> identify_interface_logic -auto_ignore \
-latch_level 1
```

SEE ALSO

```
get_ilm_objects(2)
write_ilm_netlist(2)
write_ilm_parasitics(2)
write_ilm_script(2)
write_ilm_sdf(2)
ilm_ignore_percentage(3)
```

import_power_management_mtcmos_locations

Imports the PML file for the specified cell.

SYNTAX

```
status import_power_management_mtcmos_locations
   -cell cell_name
   -library library_name
   -input_file pml_file_name
```

Data Types

```
string cel_name
string library_name
string pml_file_name
```

ARGUMENTS

-cell *cel_name*

Specifies the name of the Milkyway CEL design for which the PML file is to be imported.

-library *library_name*

Specifies the Milkyway design library in which the Milkyway CEL view exists.

-input_file *pml_file_name*

Specifies the name of PML file to be imported.

DESCRIPTION

This command imports the PML file for the specified cell design when you have edited the settings, such as on-resistance values, in the PML file. Export the PML file by using the **export_power_management_mtcmos_locations** command.

EXAMPLES

The following example imports the pml.txt file for the cell design called PMCell.

```
pr_shell>import_power_management_mtcmos_locations -library MyLib -cell PMCell -input_file pml.txt
1
```

SEE ALSO

```
export_power_management_mtcmos_locations(2)
remove_power_management_mtcmos_locations(2)
```

index_collection

Given a collection and an index into it, if the index is in range, create a new collection containing only the single object at the index in the base collection. The base collection remains unchanged.

SYNTAX

```
collection index_collection
   collection1
   index
```

Data Types

collection1	collection
index	index

ARGUMENTS

collection1

Specifies the collection to be searched.

index

Specifies the index into the collection.

The index is either a simple integer to specify the offset from the start of the collection or the string "end" optionally followed by a negative integer to specify the offset from the end of the collection.

For the start offset case the value must range from 0 to **sizeof_collection** - 1. For the end offset case the negative integer (if specified) must range from 0 to -(**sizeof_collection** - 1).

DESCRIPTION

You can use the **index_collection** command to extract a single object from a collection. The result is a new collection containing only that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index.

The range of indices is from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection1* argument. However, by definition, any index into the empty collection is invalid. Therefore, using the **index_collection** command with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

EXAMPLES

The following examples from PrimeTime use the **index_collection** command to extract the first, second, last and penultimate object of a collection.

```
pt_shell> set c1 [get_cells {u1 u2 u3}]
{"u1", "u2", "u3"}
pt_shell> query_objects [index_collection $c1 0]
{"u1"}
pt_shell> query_objects [index_collection $c1 1]
{"u2"}
pt_shell> query_objects [index_collection $c1 end]
{"u3"}
pt_shell> query_objects [index_collection $c1 end-1]
{"u2"}
```

SEE ALSO

collections(2)
query_objects(2)
sizeof_collection(2)

infer_switching_activity

Infers the switching activity on the drivers of the preset and clear pins of the current design.

SYNTAX

```
string infer_switching_activity
   [-apply]
   [-output file_name]
   [-nosplit]
   [-verbose]
```

Data Types

file_name string

ARGUMENTS

-apply

Applies the proposed switching activity annotation on the drivers of the preset and clear pins.

-output *file_name*

Writes the script to the specified file in ASCII format.

-nosplit

Prevents line splitting when the information exceeds the specified column width. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column's width, the next field begins on a new line starting with the correct column.

-verbose

Displays detailed information about the specified preset or clear pins. The current and the proposed switching activity annotation reported is applicable only to the drivers of the corresponding preset and clear pins.

DESCRIPTION

Use the **infer_switching_activity** command to infer the switching activity on the drivers of the preset and clear pins. These pins can be asynchronous preset and clear, or synchronous preset and clear. The switching activity inlcudes simple toggle rate and static probability on the drivers of these pins.

Use the **-apply** option to apply the inferred switching activity values on the drivers of the pins. When you run the

update_power command, the PrimeTime PX tool uses the inferred values to calculate and report power consumption.

For statistics on the switching activity annotation on the current design, use the **report_switching_activity** command.

EXAMPLES

In the following example, the **infer_switching_activity** command is specified after reading in the design. This command displays the current and proposed switching activity annotation for the rst pin.

```
pt_shell> infer_switching_activity
Created by infer_switching_activity on Thu May  2 17:51:45 2013

--------------------------------------------------------------------------------
                                  Current      Current    Proposed     Proposed
                                  Static       Toggle     Static       Toggle
Objects            Type           Probability  Rate       Probability  Rate
--------------------------------------------------------------------------------
rst                driver         None         None       1.00         0.00
--------------------------------------------------------------------------------
1
```

Use the **infer_switching_activity -verbose** command to display information about the receiver pin and the receiver pin types.

```
pt_shell> infer_switching_activity -verbose
Created by infer_switching_activity on Thu May  2 17:51:45 2013

--------------------------------------------------------------------------------
        Current     Current Proposed    Proposed
        Static      Toggle  Static      Toggle                Receiver
Driver  Probability Rate    Probability Rate    Receiver      Type
--------------------------------------------------------------------------------
rst     None        None    1.00        0.00    cnt_reg[0]/RN async_clear_preset
                            1.00        0.00    cnt_reg[3]/RN async_clear_preset
                            1.00        0.00    cnt_reg[1]/RN async_clear_preset
                            1.00        0.00    cnt_reg[4]/RN async_clear_preset
                            1.00        0.00    cnt_reg[2]/RN async_clear_preset
--------------------------------------------------------------------------------
There are 5 receivers for rst.
1
```

SEE ALSO

```
set_switching_activity(2)
reset_switching_activity(2)
report_power(2)
```

insert_buffer

Inserts a buffer at one or more pins.

SYNTAX

```
string insert_buffer
   [-libraries lib_spec]
   [-inverter_pair]
   [-new_net_names new_net_names]
   [-new_cell_names new_cell_names]
   pin_or_port_list
   lib_cell
```

Data Types

lib_spec	list
new_net_names	list
new_cell_names	list
pin_or_port_list	list
lib_cell	string

ARGUMENTS

-libraries *lib_spec*

If this option is specified, PrimeTime resolves the *lib_cell* value from the libraries contained only in the *lib_spec*. Libraries are searched in the order in which they appear in the *lib_spec*. *lib_spec* can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the **get_libs** command. You cannot specify this option if a full library cell name has been specified.

-inverter_pair

Indicates that a pair of inverting library cells is to be inserted instead of a single non-inverting library cell.

-new_net_names *new_net_names*

Specifies the net name to be given to the new net that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one net name. If an inverter pair is being inserted, you have to pass two net names. These names can be any valid net names, but must be the leaf names, that is, not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the **-new_cell_names** option.

-new_cell_names *new_cell_names*

Specifies the cell name to be given to the new cell that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one cell

name. If an inverter pair is being inserted, you have to pass two cell names. These names can be any valid cell names, but must be the leaf names, that is, not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the **-new_net_names** option.

pin_or_port_list

Specifies a list of pins or ports to buffer.

lib_cell

Specifies the name of the library cell to use as a buffer (or inverter if the **-inverter_pair** option is specified). The *lib_cell* option can be library cell object or the name of a library cell. The former can be obtained using the **get_lib_cells** command. The latter can either be the full library cell name, such as *lib_name/lcell_name* or the just the base name of the library cell, such as *lcell_name*. You cannot specify the **-libraries** option and explicitly specify the full library cell name. If you invoke PrimeTime with the **-multi_scenario** option, you must use only the library cell-base name. For more information, see the "RESOLVING LIBRARY CELLS" section.

DESCRIPTION

The **insert_buffer** command adds a buffer at one or more specified pins or ports. Like all other netlist editing commands, for the **insert_buffer** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The result of the **insert_buffer** command is a collection of the newly inserted cells on success, or the empty collection (empty string) on failure.

A library cell is a buffer if it has a single input and any number of outputs, as long as for each output, the logic function is either input or !input.

Newly created cells are given a unique name beginning with "U" and ending with a number; newly created nets are given a unique name beginning with "net" and ending with a number.

If you do not want PrimeTime to generate automatic names, you can use **-new_net_names** and *new_cell_names* options to override.

The **insert_buffer** command uses the following basic rules to check its arguments:

- Each pin/port must be in scope (at or below the current instance). For a description of some special scoping rules, see the "BUFFERING INSIDE BOUNDARY PINS" section.
- Each pin/port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The *lib_cell* cannot be sequential.
- The *lib_cell* must be a buffer, as previously defined.
- Without the **-inverter_pair** option, the library cell must have a noninverting output. The first noninverting output is used.
- When the **-inverter_pair** option is specified, the library cell must have an inverting output. The first inverting output is used. In this case, two cells are inserted, preserving the logic of the path.
- This command is a rare case where the objects argument (in this case, *pin_or_port_list*) does not necessarily stand alone. Pins on the same net can be grouped.

  For more information about grouping pins on the same net and for additional connection rules, see the

"CONNECTING THE NEW BUFFER" section.

Although you can mimic buffer insertion using other commands, such as the **create_cell**, **create_net**, **disconnect_net**, and **connect_net** commands, it is much more efficient to use the **insert_buffer** command.

Connecting the New Buffer

The new buffer is connected according to the following rules:

- The output of the new cell is always the new net, and the input of the new cell is always the old net. For example, buffering e1/Z or e3/A as follows:

```
e1/Z --- old_net --- e3/A

creates the following:

e1/Z --- old_net --- U1 --- net1 --- e3/A
```

- If the pin is a load, it is disconnected from its net, and connected to the new net.
- If the pin is a driver, all loads on that net are disconnected from the old net and connected to the new net.
- When pins on the same net are specified, they are grouped, and one buffer is inserted. Loads and drivers are grouped separately.
- If the net is multidriven, all driving pins must be specified for the command to succeed. In the following example, you cannot specify only e1/Z; you must specify both e1/Z and e2/Z. However, for the loads in Circuit 1, any combination can be buffered.

```
e1/Z ---+                +--- e3/A
        |                |
        +----- net ------+--- e4/A
        |                |
e2/Z ---+                +--- e5/A

            Circuit 1
            ---------
```

- Certain parallel buffer cases are also examined more closely. In Circuit 2, you cannot subdivide the inputs of the parallel drivers. You must buffer the inputs of both cell1 and cell2.

```
        +--- cell1 ---+
net1 ---+             +-- net2 ---
        +--- cell2 ---+

             Circuit 2
             ---------
```

BUFFERING INSIDE BOUNDARY PINS

When you specify insertion of a buffer at a pin on the boundary of a hierarchical block, the **insert_buffer** command inserts the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To insert the buffer inside the hierarchical block, set the scope to that block using the **current_instance** command and then execute the **insert_buffer** command.

The buffer is inserted within the block to which the **current_instance** command is set. For an example of buffering inside boundary pins, see the EXAMPLES section.

Buffering and its Effects on Parasitics

When parasitics are present and buffer insertion is performed the parasitics will only be preserved under the following conditions. (Note: What applies for a single buffer also applies for an inverter pair)

Buffering load pins in the presence of parasitics.

When a single load pin connected to a net with parasitics is buffered, the parasitics are preserved on the net connected to the input of the inserted buffer. There are no parasitics on the net connected to the output of the inserted buffer.

Buffer load pin e3/A as follows:

```
                     +-- e4/A
                     |
e1/Z --- old_net ---|-- e3/A
            ^        |
            |        +-- e5/A
            |
      parasitics here

results in the parasitics being kept on the old net

                     +-- e4/A
                     |
e1/Z --- old_net ---|-- U1 --- net1 --- e3/A
            ^        |            ^
            |        +-- e5/A     |
            |                     |
      parasitics here     no parasitics
```

Buffering more than a single load pin results in the parasitics on the net connected to the input of the inserted buffer being removed and a warning being issued. (See PARA-060)

```
Buffer load pins e3/A, e4/A and e5/A

                     +-- e4/A
                     |
e1/Z --- old_net ---|-- e3/A
            ^        |
            |        +-- e5/A
            |
      parasitics here

results in the parasitics being removed

                                      +-- e4/A
                                      |
e1/Z --- old_net --- U1 --- net1 ---|-- e3/A
            ^                ^        |
            |                |        +-- e5/A
            |                |
     no parasitics     no parasitics
```

Buffering driver pins in the presence of parasitics.

When a single driver pin is buffered, on a net with only one driver, the parasitics are moved to the net connected to the output of the inserted buffer. There are no parasitics on the net connected to the input of the inserted buffer.

```
Buffer driver pin e1/Z as follows

                     +-- e4/A
                     |
```

```
e1/Z --- old_net ---|-- e3/A
            ^       |
            |       +-- e5/A
            |
      parasitics here

results in the parasitics being moved to net1

                                    +-- e4/A
                                    |
e1/Z --- old_net --- U1 --- net1 ---|-- e3/A
            ^                ^      |
            |                |      +-- e5/A
            |                |
     no parasitics     parasitics move here
```

Buffering more than a single driver pin on a multiply driven net results in the parasitics being removed from the net connected to the output of the inserted buffer. (See PARA-060)

Buffer driver pin e1/Z, e2/Z and e6/Z as follows:

```
e2/Z --+               +-- e4/A
       |               |
e1/Z --|--- old_net ---|-- e3/A
       |       ^       |
e6/Z --+       |       +-- e5/A
               |
       parasitics here

results in the parasitics being removed

e2/Z --+                             +-- e4/A
       |                             |
e1/Z --|--- old_net --- U1--- net1 ---|-- e3/A
       |       ^               ^      |
e6/Z --+       |               |      +-- e5/A
               |               |
        no parasitics    no parasitics
```

Resolving Library Cells

If the *lib_cell* option has been specified in base name only format, such as without a library from which to resolve it from, PrimeTime resolves the library cell according to the following methodology:

- If the **-libraries** option is specified, PrimeTime searches for library cells in the libraries contained within the *lib_spec* only.
- Alternatively, if the **-libraries** option is not specified, PrimeTime searches for the library cell in the libraries contained within the **link_path** variable.

The first library cell that is found is used.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with

swap_cell. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

Refer to Circuit 1 for the following example; e1/Z and e2/Z both drive *net*. The first command specifies a *lib_cell* value that is not a buffer; the command fails, and an error message is generated. The second command specifies a buffer for a *lib_cell*, but does not specify all driver pins on a multi-driven net. Again the command fails with an error message. The third command specifies a buffer for *lib_cell* and specifies all pins for the multi-driven net. This time the command succeeds, and creates the new cell U1 and the new net net1.

```
pt_shell> insert_buffer e1/Z class/AN2P
Error: Could not insert 'class/AN2P' -
    lib_cell is not a buffer. (NED-010)
Error: No changes made. (NED-040)

pt_shell> insert_buffer e1/Z class/B1I
Error: Could not insert a buffer on object 'e1/Z':
    Multi-driver net and not all driver pins specified. (NED-012)
Error: No changes made. (NED-040)

pt_shell> insert_buffer {e1/Z e2/Z} class/B1I
Information: Inserted 'U1' at 'e1/Z', 'e2/Z' with 'class/B1I'. (NED-046)
{"U1"}
```

Refer to Circuit 1 for the following example. In the first command, an inverting buffer is inserted on two of the loads of "net". Two new cells are created, U2 and U3, as well as two new nets, net2 and net3. The second and third commands show the use of the **report_cell** and **report_net** commands to display the connections. Excerpts from the reports are shown.

```
pt_shell> insert_buffer {e3/A e4/A} class/IV -inverter_pair
Information: Inserted 'U2' and 'U3' at 'e3/A', 'e4/A' with 'class/IV. (NED-046)
{"U2", "U3"}
pt_shell> report_cell -connections e4
****************************************
Report : cell
         -connections
****************************************

Connections for cell 'e4':
    Reference:          B1I
    Library:            class

    Input Pins          Net
    ----------------    ------------
    A                   net3

    Output Pins         Net
    ----------------    ------------
    Z                   out2

1

pt_shell> report_net -connections net3
****************************************
Report : net
         -connections
****************************************

Connections for net 'net3':

    Driver Pins              Type
    ------------             ----------------
    U3/Z                     Output Pin (IV)

    Load Pins                Type
    ------------             ----------------
    e4/A                     Input Pin (B1I)
    e3/A                     Input Pin (B1I)

1
```

The following example demonstrates buffer insertion inside the boundary of a hierarchical block. To buffer a boundary pin on the inside of the hierarchical block, you must set the current instance to that block, as shown. Otherwise, the **insert_buffer** command would insert the buffer at the top level, outside instance e1.

Note the method used to get the boundary pin from within the block.

```
pt_shell> current_instance e1
u1
pt_shell> insert_buffer [get_pins ./z1] class/B1I
Uniquifying 'e1' (UBLOCK) as 'UBLOCK_0'.
Information: Inserted 'U1' at 'e1/z1' with 'class/B1I'. (NED-046)
{"e1/U1"}
```

SEE ALSO

```
current_instance(2)
get_pins(2)
remove_buffer(2)
report_cell(2)
report_net(2)
size_cell(2)
swap_cell(2)
write_changes(2)
link_path(3)
PARA-060(n)
```

insert_virtual_decap_cells

Insert virtual decap cells to the specified region.

SYNTAX

```
status insert_virtual_decap_cells
   -total_capacitance cap_value
   -bounding_box rectangle
   -connect_to_power  power_net_name
   -connect_to_ground ground_net_name
```

Data Types

```
float  cap_value
list   rectangle
string power_net_name
string ground_net_name
```

ARGUMENTS

-total_capacitance *cap_value*

Specifies total capacitance of virtual decap cells to be inserted to the region. The cap_value needs to be specified in library unit.

-bounding_box *rectangle*

Specifies the rectangular region where virtual decap cells are to be inserted. The rectangle is specified as a list of four coordinates, {left bottom right top}. The coordinates need to be specified in micron unit.

-connect_to_power *power_net_name*

Specifies the power supply net to which to connect power pins of the virtual decap cells.

-connect_to_ground *ground_net_name*

Specifies the ground supply net to which to connect ground pins of the virtual decap cells.

DESCRIPTION

The **insert_virtual_decap_cells** command inserts virtual decap cells to the specified region. For each existing standard (logic, filler, or decap) cell in the specified region, one virtual decap cell is created and attached to it. The total capacitance value of the virtual decap cells created is the specified *cap_value*. The capacitance value of

all virtual decap cells created is equal. When the calculate_rail_voltage command is invoked, the capacitance of each virtual decap cell is inserted between the power and ground pins of the standard cell it is attached to. For each standard cell with a virtual decap cell attached,

1. If both the power and ground pins of the standard cell are connected to the power and ground supply nets identical to those of the virtual decap cell, the capacitance is inserted between the power and ground pins of the standard cell.
2. If only the power pin of the standard cell is connected to the power supply net identical to that of the virtual decap cell, the capacitance is inserted between the power pin of the standard cell and the true ground.
3. If only the ground pin of the standard cell is connected to the ground supply net identical to that of the virtual decap cell, the capacitance is inserted between the ground pin of the standard cell and the true ground.
4. If none of the power and ground pins of the standard cell are connected to the power and ground supply nets identical to those of the virtual decap cell, the capacitance is ignored.

When one or more insert_virtual_decap_cell commands are invoked in the same session, a standard cell which already has a virtual decap cell attached may be selected by the command. When this happens,

1. If the power and ground supply nets of the virtual decap cell match those specified in the current **insert_virtual_decap_cell** command, the capacitance value of the virtual decap cell is increased by the amount specified in the current command.
2. If either the power or ground supply net of the virtual decap cell does not match those specified in the current **insert_virtual_decap_cell** command, the capacitance value of the virtual decap cell is overwritten by the new amount specified in the current command.

EXAMPLE

The following example inserts virtual decap cells to the specified region, {100 100 1000 1000}. The power pins of the virtual decap cells are connected to the power supply net VDD and the ground pins are connected to the ground supply net VSS. The total capacitance of 1pF is equally split among the virtual decap cells being created where the library capacitance unit is pF.

```
pr_shell> insert_virtual_decap_cells -total_capacitance 1 \
          -region {100 100 1000 1000} \
          -connect_to_power VDD -connect_to_ground VSS
1
```

The following example reports the virtual decap cells inserted between the power supply net VDD and ground supply net VSS by the previous insert_virtual_decap_cells command.

```
pr_shell> report_virtual_decap_cells -connect_to_power VDD -connect_to_ground VSS
****************************************
Report : Virtual decap cells inserted
Design : design
Version: J-2014.06-alpha
Date   : Mon Mar 24 15:07:45 2014
****************************************

Coupled power domains: VDD VSS

Format: instance_name virtual_decap_value [pF]
INV11      0.077 [pF]
ARCR0_1    0.077 [pF]
INV21      0.077 [pF]
INV31      0.077 [pF]
INV41      0.077 [pF]
FEED0_14   0.077 [pF]
INV51      0.077 [pF]
FEED0_5    0.077 [pF]
```

```
INV61      0.077 [pF]
FEED0_4    0.077 [pF]
FEED0_3    0.077 [pF]
FEED0_2    0.077 [pF]
FEED0_1    0.077 [pF]

Total number of virtual decap cells in the coupled power domains: 13
Total decoupling capacitance inserted to the coupled power domains:      1.000 [pF]
1
```

SEE ALSO

```
report_virtual_decap_cells(2)
remove_virtual_decap_cells(2)
calculate_rail_voltage(2)
```

is_false

Tests the value of a specified variable, and returns 1 if the value is 0 or the case-insensitive string **false**; returns 0 if the value is 1 or the case-insensitive string **true**.

SYNTAX

```
status is_false
   value
```

Data Types

```
value      string
```

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 0 or the case-insensitive string **false**. The command returns 0 if the value is either 1 or the case-insensitive string **true**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_false** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE
if { !$x } {
  set y TRUE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_false** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_false** command:

```
prompt> set x TRUE
TRUE

prompt> if { ![is_false $x] } {
?         set y TRUE
?       }
TRUE

prompt>
```

SEE ALSO

expr(2)
if(2)
is_true(2)

is_true

Tests the value of a specified variable, and returns 1 if the value is 1 or the case-insensitive string **true**; returns 0 if the value is 0 or the case-insensitive string **false**.

SYNTAX

```
status is_true
   value
```

Data Types

```
value      string
```

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 1 or the case-insensitive string **true**. The command returns 0 if the value is either 0 or the case-insensitive string **false**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_true** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE
if { !$x } {
  set y FALSE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_true** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_true** command:

```
prompt> set x FALSE
FALSE

prompt> if { ![is_true $x] } {
?         set y FALSE
?       }

FALSE

prompt>
```

SEE ALSO

expr(2)
if(2)
is_false(2)

license_users

Lists the current users of the Synopsys licensed features.

SYNTAX

```
string license_users
   [feature_list]
```

Data Types

```
feature_list     list
```

ARGUMENTS

feature_list

Specifies a list of licensed features for which to obtain the information. If you do not use this option, all features are shown.

DESCRIPTION

This command displays information about all licenses, related users, and host names currently in use. This command works only when Network Licensing is enabled.

EXAMPLES

In this example, all users of the PrimeTime feature are displayed.

```
pt_shell> license_users PrimeTime

john@node2     PrimeTime
paul@node1     PrimeTime
george@node3   PrimeTime, PrimeTime
rstarr@node3   PrimeTime

4 users listed.
```

SEE ALSO

```
get_license(2)
list_licenses(2)
remove_license(2)
```

link_design

Resolves references in a design.

SYNTAX

```
string link_design
   [-verbose]
   [-force]
   [-remove_sub_designs]
   [-keep_sub_designs]
   [design_name]
```

Data Types

```
design_name     string
```

ARGUMENTS

-verbose

Displays verbose messages.

-force

Forces relinking of the design. If you omit this option, the tool does not relink a fully linked design.

-remove_sub_designs

Removes subdesigns after linking. Use this option to free up memory and improve performance. For more information, see the "Performance Considerations" section.

-keep_sub_designs

Keeps subdesigns after linking. By default, subdesigns are removed. Use this option to keep the subdesigns in memory so that the current design can be changed to other designs later.

design_name

Specifies the design to be linked; the default is the current design.

DESCRIPTION

The **link_design** locates all designs and library components that are referenced by the current design and links them to the current design. During linking, the tool loads all files specified by the **link_path** variable if they are not

already in memory. Successful linking results in a fully instantiated design on which you can perform analysis.

By default, the case sensitivity of the link is determined by the source of the objects being linked. Although it is not recommended, you can change the default behavior by setting the **link_force_case** variable.

Automatic Loading of Designs and Libraries

If you set the **link_path** and **search_path** variables, you need to read only your top-level design and then link. The tool automatically finds and loads all other required designs and libraries.

In the following example, the newcpu.db top-level design uses the cmos.db library. Since the **link_path** variable specifies cmos.db, the **link_design** command loads cmos.db. As linking proceeds, the tools loads any required design that is not already in memory by searching the paths specified by the **search_path** variable. For example, the referenced BOX1 design is not in memory, so the tool searches for BOX1.db in the **search_path** variable and loads it.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos

pt_shell> set link_path "* cmos.db"
* cmos.db

pt_shell> read_db newcpu.db
Loading db file '/designs/newcpu/v1.6/dbs/newcpu.db'
1

pt_shell> link_design newcpu
Loading db file '/libs/cmos/cmos.db'
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Design 'newcpu' was successfully linked.
1
```

Automatic Linking

Many PrimeTime commands attempt to link the current design for you. For example, the **report_timing** command performs linking if the current design is not linked. Automatic linking occurs only if the design is completely unlinked, and not if the current design is partially linked and has unresolved references. If the current design is fully linked, there is no need for automatic linking, so it is not attempted.

You can disable automatic linking by setting the **auto_link_disable** variable to **true**. Because determining the need for linking takes little runtime, setting this variable is normally unnecessary. The variable provides compatibility with Design Compiler.

Unresolved References

If the **link_design** command fails to resolve one or more references, you need to correct the source of the problem and relink the design. Failures are typically caused by

- Missing libraries or designs
- Incorrectly specified **link_path** or **search_path** variable
- A file that is located in the path but not accessible by you

After making the necessary corrections, you can decide whether to do an incremental relink or an initial link, or whether to allow the too to create black boxes for the unresolved references.

If you can resolve the references by changing the **link_path** or **search_path** variable, you must relink the

design.

The creation of black boxes is controlled by the **link_create_black_boxes** variable, which is set to **true** by default. Unless you set this variable to **false**, the tool automatically converts each unresolved reference to a black box (an empty cell with no timing arcs). The result is a completely linked design on which analysis can be performed (assuming there are no other unrecoverable link errors). To prevent unresolved references, set the **link_create_black_boxes** variable to **true**.

Black box creation can fail when there are multiple conflicting references, usually with generic logic. For example, if SELECT_OP has two references, one with five pins and the other with 20 pins, the second black box might not be created, and the design might not link. PrimeTime does not support generic logic, except for GTECH; if a design contains such generic logic, remove the design or replace it with an empty design.

Sometimes, black box creation fails. This occurs when there are multiple conflicting references. This happens most often with generic logic. For example, if there are two references to SELECT_OP, one with five pins, and the other with 20 pins, it is likely that the second black box is not created and the design does not link. Other than GTECH, PrimeTime does not support generic logic. Designs containing such generic logic should be removed or replaced by empty designs.

Linking With Mismatches

By default, if there are pin mismatches between instance and reference, the **link_design** command issues errors and fails. If you set the **link_allow_design_mismatch** variable to **true**, the **link_design** command issues warnings and continues. This allows you to gather useful information even when part of a design is missing.

To report mismatches, use the **report_design_mismatch** command.

Performance Considerations

The tool starts linking with the top design along with any other loaded designs and libraries. During the link process, other necessary designs and libraries are loaded. When the link process completes successfully, it produces a design that can be analyzed.

By default, all subdesigns used to build the linked design are removed from memory. Use the **-keep_sub_designs** option to retain the subdesigns if you want to

- Keep all designs and analyze them later
- Relink the design later with the **-force** option

Note that this option takes more memory.

You need subdesigns only if you want to analyze more than one design in a session; the linked design contains all information necessary for analysis. For example, in PrimeTime, wire load models are set on specific instances of the design, without requiring a subdesign.

EXAMPLES

The following examples show how a design links with many of the different linker options. First, the link fails when a library cannot be found in the link path because of a typo in the search path.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmoss"
/designs/newcpu/v1.6/dbs /libs/cmoss
pt_shell> set link_path "* cmos.db"
* cmos.db
```

```
pt_shell> read_db newcpu.db
Loading db file '/design/newcpu/v1.6/dbs/newcpu.db'
1
pt_shell> link_design newcpu
Error: Can't read link_path file 'cmos.db' (LNK-001)
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Warning: Unable to resolve reference to 'NR4' in 'newcpu'. (LNK-005)
Warning: Unable to resolve reference to 'ND2' in 'newcpu'. (LNK-005)
Warning: Unable to resolve reference to 'FD2' in 'newcpu'. (LNK-005)
Information: Design 'newcpu' was not successfully linked:
             16 unresolved references. (LNK-003)
```

Correcting the value of the **search_path** variable and doing an initial link completely links the design without black boxes.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos
pt_shell> link_design newcpu
Loading db file '/libs/cmos/cmos.db'
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Design 'newcpu' was successfully linked.
1
```

SEE ALSO

current_design(2)
list_designs(2)
list_libs(2)
read_db(2)
read_ddc(2)
read_verilog(2)
read_vhdl(2)
report_design_mismatch(2)
auto_link_disable(3)
link_allow_design_mismatch(3)
link_create_black_boxes(3)
link_force_case(3)
link_path(3)
search_path(3)

list_attributes

Lists currently defined attributes.

SYNTAX

```
string list_attributes
   [-application]
   [-class class_name]
   [-nosplit]
```

Data Types

```
class_name       string
```

ARGUMENTS

-application

Lists application attributes as well as user-defined attributes.

-class *class_name*

Limit the listing to attributes of a single class. Valid classes are design, port, cell, net, and so on.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **list_attributes** command displays an alphabetically sorted list of currently defined attributes. PrimeTime attributes divide into two categories: application-defined and user-defined. By default, the **list_attributes** command lists all user-defined attributes.

Using the **-application** option adds all application attributes to the listing. Note that there are many application attributes. It is often useful to limit the listing to a specific object class using the *class_name*.

EXAMPLES

This is an example listing of some attributes defined with **define_user_attribute**.

```
pt_shell> list_attributes

****************************************
Report : List of Attribute Definitions
Design :
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from db (for user-defined)

Attribute Name   Object  Type      Properties  Constraints
-------------------------------------------------------------
attr_b           cell    boolean   U
attr_d           cell    double    U
attr_f           cell    float     U
attr_i           cell    int       U,I
attr_ir1         cell    int       U           0 to 100
attr_ir2         cell    int       U           >= 0
attr_ir3         cell    int       U           <= 100
attr_oo          cell    string    U           A, B, C, D
attr_s           cell    string    U
attr_s           net     string    U
```

This example limits the listing to net attributes only.

```
pt_shell> list_attributes -application -class net

****************************************
Report : List of Attribute Definitions
Design :
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from db (for user-defined)

Attribute Name         Object  Type      Properties  Constraints
-------------------------------------------------------------------
area                   net     float      A
attr_s                 net     string     U
ba_capacitance_max     net     float      A
ba_capacitance_min     net     float      A
ba_resistance_max      net     float      A
ba_resistance_min      net     float      A
base_name              net     string     A
full_name              net     string     A
net_resistance_max     net     float      A
net_resistance_min     net     float      A
object_class           net     string     A
pin_capacitance_max    net     float      A
pin_capacitance_min    net     float      A
total_capacitance_max  net     float      A
total_capacitance_min  net     float      A
```

```
wire_capacitance_max   net     float      A
wire_capacitance_min   net     float      A
```

SEE ALSO

```
define_user_attribute(2)
get_attribute(2)
remove_user_attribute(2)
report_attribute(2)
set_user_attribute(2)
```

list_designs

Lists designs that have been read into PrimeTime.

SYNTAX

```
string list_designs
   [-all]
   [-only_used]
```

ARGUMENTS

-all

In addition to listing the designs in memory, the **-all** option lists the designs that are instantiated in the current design but are removed from memory.

-only_used

Lists only designs in use, including the current design, any design that is linked or partially linked, and any designs instantiated in a linked design.

DESCRIPTION

The **list_designs** command lists the designs that have been read into PrimeTime. When used without options, the commands lists only designs that are currently in memory.

The command output shows the status of each design with the following notation:

* - Design is the current design

L - Design is linked

N - Design is not in memory

l - Design is partially linked

EXAMPLES

The following example lists the designs in memory.

```
pt_shell> list_designs

Design Registry:
 *L AD4FULA        /abc/designs/top.db:AD4FULA
  N AD4PG          /abc/designs/add1.db:AD4PG
    ADD5A          /abc/designs/add2.db:ADD5A
    MULT1          /abc/designs/top.db:MULT1
```

The following example lists only the designs that are in use and in memory.

```
pt_shell> list_designs -only_used
Design Registry:
 *L AD4FULA        /abc/designs/top.db:AD4FULA
    ADD5A          /abc/designs/add2.db:ADD5A
    MULT1          /abc/designs/top.db:MULT1
```

If you remove the MULT1 design, you can display it using the **-all** option only.

```
pt_shell> remove_design MULT1
Removing design 'MULT1'...
1
pt_shell> list_designs -all -only_used
Design Registry:
 *L AD4FULA        /abc/designs/top.db:AD4FULA
    ADD5A          /abc/designs/add2.db:ADD5A
  N MULT1          /abc/designs/top.db:MULT1
```

SEE ALSO

current_design(2)
link_design(2)
remove_design(2)

list_key_bindings

Displays all the key bindings and edit mode of current shell session.

SYNTAX

```
status list_key_bindings
   [-nosplit]
```

ARGUMENTS

-nosplit

Indicates that lines are not to be split when column fields overflow.

DESCRIPTION

The **list_key_bindings** command displays current key bindings and the edit mode. To change the edit mode, set the **line_editing_mode** variable in either the .synopsys_pt.setup file or directly in the shell.

The text CTRL+K is read as `Control+K' and describes the character produced when the k key is pressed while the Control key is depressed.

The text META+K is read as `Meta+K' and describes the character produced when the Meta key is depressed, and the k key is pressed. The Meta key is labeled ALT on many keyboards. On keyboards with two keys labeled ALT (usually to either side of the space bar), the ALT on the left side is generally set to work as a Meta key. The ALT key on the right can also be configured to work as a Meta key or can be configured as some other modifier, such as a Compose key for typing accented characters.

If you do not have a Meta key, ALT key, or another key working as a Meta key, the identical keystroke can be generated by typing ESC first and then typing k. Either process is known as metafying the k key.

Alternative key bindings work only in vi alternate (command) mode.

SEE ALSO

sh_enable_line_editing(3)
sh_line_editing_mode(3)

list_libs

Lists libraries that are read into PrimeTime.

SYNTAX

```
string list_libs
   [-only_used]
```

ARGUMENTS

-only_used

Lists only the libraries in use. A library is in use if a linked design links to library cells from the library.

DESCRIPTION

The **list_libs** command lists the libraries that are read into PrimeTime. You can filter unused libraries out of the display by using the **-only_used** option. A library is in use if a linked design links to library cells from the library.

The command output shows the status of each library with the following notation:

* - Main library for the current design. The main library is the first library in the link path which has a time unit.

M - Maximum library of a maximum/minimum library relationship created by the **set_min_library** command.

m - Minimum library of a maximum/minimum library relationship created by the **set_min_library** command.

EXAMPLES

The following example lists all libraries.

```
pt_shell> list_libs

Library Registry:
    bus1_lib      /u/lib/bus1_lib.db:bus1_lib
    tech1         /u/lib/tech1.db:tech1
```

After linking a design, the main library can be identified. Further, using the **-only_used** option limits the display to the libraries that were used to link the current design.

```
pt_shell> link_design top_flat
Linking design top_flat...

Designs used to link top_flat:
  <None>

Libraries used to link top_flat:
  tech1

Design 'top_flat' was successfully linked.
1
pt_shell> list_libs -only_used

Library Registry:
 *  tech1         /u/lib/tech1.db:tech1
```

SEE ALSO

current_design(2)
link_design(2)
list_designs(2)
set_min_library(2)

list_licenses

Shows the licenses that are currently checked out.

SYNTAX

status **list_licenses**

ARGUMENTS

None.

DESCRIPTION

The **list_licenses** command lists the name and number of licenses that you currently have checked out.

In distributed multi-scenario analysis (DMSA), the command issued at the master process returns the total count of licenses checked out for the analysis. If you run the command at a worker process, the command shows the licenses in use by the current scenario running at the worker process.

If scenarios are running sequentially because of a limited number of worker processes or licenses, running **list_licenses** at a worker process shows the instantaneous license count at the moment of command execution.

EXAMPLES

This example shows the output from the **list_licenses** command.

```
pt_shell> list_licenses

Licenses in use:
        PrimeTime (1)
        PrimeTime-SI (1)
```

SEE ALSO

```
get_license(2)
license_users(2)
remove_license(2)
```

list_mw_cels

Prints the list of Milkyway designs in the current Milkyway library.

SYNTAX

```
status list_mw_cels
   [-all_views]
   [-all_versions]
   [-sort]
```

ARGUMENTS

-all_views

List all views of Milkyway designs in the current Milkyway library. If specifying this option without **-all_versions** option, the version number will not be listed.

-all_versions

List all versions of Milkyway designs in the current Milkyway library. If specifying this option without **-all_views** option, the designs of CEL view will only be listed.

-sort

Sort the list by alphabet.

DESCRIPTION

This command prints the list of Milkyway designs in the current Milkyway library. The library must be open. And note that it will show the design name if any view of the design exists in the library. It also prints the following information about the Milkyway design:

- **open** if it is an open Milkyway design.
- **read_only** if it is opened in read only mode.
- **current** if it is the currently active Milkyway design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example lists the Milkyway designs in the current library:

```
prompt> list_mw_cels
top
AND
NOR
1
prompt> list_mw_cels -all_views
top.CEL
AND.CEL
NOR.CEL
NOR.FRAM
1
prompt> list_mw_cels -all_versions
top.CEL;1
top.CEL;2
AND.CEL;1
NOR.CEL;1
1
prompt> list_mw_cels -all_views -all_versions -sort
AND.CEL;1
NOR.CEL;1
NOR.FRAM;1
top.CEL;1
top.CEL;2
1
```

SEE ALSO

get_mw_cels(2)

list_sidefiles

Lists all PrimeRail library characterization sidefiles that have been read into the session.

SYNTAX

```
string list_sidefiles
```

ARGUMENTS

None

DESCRIPTION

Lists all of the PrimeRail library characterization sidefiles that have been read into the session. This includes the sidefiles found through **rail_sidefile_path** and the sidefiles that have been loaded directly from Milkyway when the **rail_autoload_mw_sidefile** variable is set to *true*.

EXAMPLES

The following example lists sidefiles read from the Milkyway design:

```
pr_shell> set rail_autoload_mw_sidefile true
pr_shell> open_mw_cel -lib my_lib my_cel
pr_shell> list_sidefiles

Rail analysis library characterization sidefiles:
my_lib/lib_21
my_lib/lib_23
```

The following example lists sidefiles read from rail_sidefile_path:

```
pr_shell> set rail_sidefile_path { my.cw my.cin }
pr_shell> open_mw_cel -lib my_lib my_cel
pr_shell> list_sidefiles

Rail analysis library characterization sidefiles:
my.cw
my.cin
```

SEE ALSO

```
rail_autoload_mw_sidefile(3)
rail_sidefile_path(3)
```

lminus

Removes one or more named elements from a list and returns a new list.

SYNTAX

```
list lminus
   [-exact]
   original_list
   elements
```

Data Types

```
original_list     list
elements          list
```

ARGUMENTS

-exact

Specifies that the exact pattern is to be matched. By default, **lminus** uses the default match mode of **lsearch**, the **-glob** mode.

original_list

Specifies the list to copy and modify.

elements

Specifies a list of elements to remove from *original_list*.

DESCRIPTION

The **lminus** command removes elements from a list by using the element itself, rather than the index of the element in the list (as in **lreplace**). The **lminus** command uses the **lsearch** and **lreplace** commands to find the elements and replace them with nothing.

If none of the elements are found, a copy of *original_list* is returned.

The **lminus** command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

EXAMPLES

The following example shows the use of the **lminus** command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}
a b c

prompt> set l2 [lminus $l1 {a b d}]
c

prompt> set l3 [lminus $l1 d]
a b c
```

The following example illustrates the use of **lminus** with the **-exact** option:

```
prompt> set l1 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c

prompt> set l2 [lminus $l1 a*]
{b[1]} b c

prompt> set l3 [lminus -exact $l1 a*]
a {a[1]} {b[1]} b c

prompt> set l4 [lminus -exact $l1 {a[1] b[1]} ]
a a* b c
```

SEE ALSO

lreplace(2)
lsearch(2)

load_of

Gets the capacitance of a library cell pin. It is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
float load_of
   lib_pin
```

Data Types

```
lib_pin       string
```

ARGUMENTS

lib_pin

Specifies the name of the library cell pin, or a collection that contains the library cell pin, for which to get the capacitance.

DESCRIPTION

The **load_of** command returns the capacitance of the given library cell pin. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the **load_of** command can be found in the Design Compiler documentation. The supported method for getting the capacitance of a library cell pin is by using the **get_attribute** command with the **pin_capacitance** attribute.

SEE ALSO

get_attribute(2)

load_power_management_mtcmos_locations

2012"

Loads the power management MTCMOS device location file (PML file) to the current session and override the existing PML data of the specified hard macro cell.

SYNTAX

```
status load_power_management_mtcmos_locations
   object_list
   -file file_name
   -force
   -verbose
```

Data Types

```
list object_list
string file_name
```

ARGUMENTS

object_list

Specifies a list of library cell names or a library cell collection. The library cell name list doesn't support wildcard. It's a mandatory option.

-file_name

Specifies the input file name of power management MTCMOS device location file (PML file). It's a mandatory option. The file can be exported by **export_power_management_mtcmos_locations** command in **mm_shell**.

-force

forces to load the file and skip errors. When the option isn't specified, the command doesn't load the file to memory if any error occurs.

-verbose

print detail information when parsing the given file.

DESCRIPTION

This command loads the power management MTCMOS device location file (PML file) to the current session and overrides the existing PML data of the given hard macro cell. This command will do sanity check for the input PML file. All duplicated on-state resistors and on-state resistors not connect to metal shapes will be ignored.

To enable hard macro power management cell support and use this command, the tcl variable **rail_enable_hm_pm_feature** value has to be true.

The PML file has the source and drain locations on each power management MTCMOS devices. Each power management MTCMOS device will be modeled as a resistor with a on-state resistance.

The format of PML file is <net_name1> <layer_number1> <x1> <y1> <net_name2> <layer_number2> <x2> <y2> <on-state resistance>

For example:

comment: example

```
VSS_IN 31 50150 112965 VSS_OUT 31 51470 115482 460
VSS_IN 50150 112965 VSS_OUT 31 48830 115482 460
VSS_IN 60230 112965 VSS_OUT 31 58910 115482 460
```

where the unit of coordinate is Milkyway database unit, and the unit of on-state resistance is ohm.

If a library cell loads PML file by this command before **open_mw_cel**, open_mw_cel won't load PML data from Milkyway database for the library cell, since the PML data loaded by this command has higher priority than the PML data in Milkyway database loaded by **open_mw_cel**. If the command is invoked after **open_mw_cel**, the command only loads PML file for hard macros which already have PML data before **open_mw_cel**.

A typical usage of this this command is for a what-if flow where the resistance values can be adjusted. After loading the given PML file, user can run **calculate_rail_voltage -static** right away and then quickly iterating with static rail analysis to assess result tradeoffs.

EXAMPLES

The following example loads PML file for hard macro "HM_SWITCH"

```
pr_shell> load_power_management_mtcmos_locations \
                  [get_lib_cell */HM_SWITCH] -file pml.txt

pr_shell> load_power_management_mtcmos_locations HM_SWITCH \
                  -file pml.txt
```

SEE ALSO

report_hard_macro_power_management_mtcmos(2)

ls

Lists the contents of a directory.

SYNTAX

```
string ls [filename ...]
```

```
string filename
```

ARGUMENTS

filename

Provides the name of a directory or filename, or a pattern which matches files or directories.

DESCRIPTION

If no argument is specified, the contents of the current directory are listed. For each *filename* matching a directory, **ls** lists the contents of that directory. If *filename* matches a file name, the file name is listed.

EXAMPLES

```
shell> ls *.db *.pt

test1.pt         c1.db          c3.db           c5.db

test2.pt         c2.db          c4.db           c6.db
```

SEE ALSO

cd(2)
pwd(2)

man

Displays reference manual pages.

SYNTAX

```
string man
   topic
```

Data Types

topic	string

ARGUMENTS

topic

Specifies the subject to display. Available topics include commands, variables, and error messages

DESCRIPTION

The **man** command displays the online manual page for a command, variable, or error message. You can write man pages for your own Tcl procedures and access them with the **man** command by setting the **sh_user_man_path** variable to an appropriate value. See the man page for the **sh_user_man_path** variable for details.

EXAMPLES

The following command displays the man page for the **echo** command:

```
prompt> man echo
```

The following command displays the man page for the error message CMD-025:

```
prompt> man CMD-025
```

SEE ALSO

```
help(2)
sh_user_man_path(3)
```

map_design_mode

Maps specified design modes to cell modes, paths, or both.

SYNTAX

```
status map_design_mode
   [-from from_pin_list]
   [-to   to_pin_list]
   [-through through_pin_list]
   design_mode
   [cell_mode_list]
   [instance_list]
```

Data Types

from_pin_list	list
to_pin_list	list
through_pin_list	list
design_mode	list
cell_mode_list	list
instance_list	list

ARGUMENTS

-from *from_pin_list*

Specifies a timing path from a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

-to *to_pin_list*

Specifies a timing path to a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

-through *through_pin_list*

Specifies a timing path through a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

design_mode

Specifies a design mode that is mapped to cell modes or paths. The design modes on the list must have been previously created using the **define_design_mode_group** command.

instance_list

Specifies a list of cells in which the specified modes are mapped to the design mode. This list must be accompanied by the *cell_mode_list* option.

cell_mode_list

Specifies a list of cell modes which are mapped to the design mode.

DESCRIPTION

The **map_design_mode** command maps design modes to cells or paths. The design modes must have been previously created by the **define_design_mode_group** command.

To map a design mode to a cell mode, use this command:

```
map_design_mode cell_mode_list instance_list design_mode
```

If the design mode changes from inactive to active, then all cell modes mapped to the design mode also become active.

To map a design mode to a path, use this command:

```
map_design_mode -from from_pin_list -to to_pin_list
  -through through_pin_list design_mode
```

When a path is mapped to a design mode and the design mode is set inactive, the path becomes a false path. If the design mode is subsequently set active, then the path is reset and no longer considered false.

To see a report of the design modes specified for the current design, use the **report_mode -type design** command. The report also shows the cell modes and paths mapped to each design mode.

EXAMPLES

The following example defines two design modes, DM1 and DM2, maps the cell mode READ to design mode DM1 (for instance Uram1), then removes the design mode DM1. Removing DM1 also removes the mapping to the active cell mode READ.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode READ Uram1 DM1
pt_shell> remove_design_mode DM1
```

SEE ALSO

```
define_design_mode_group(2)
map_design_mode(2)
report_mode(2)
reset_mode(2)
set_mode(2)
```

mem

Shows the total memory used by PrimeTime.

SYNTAX

```
integer mem
```

ARGUMENTS

None.

DESCRIPTION

The **mem** command shows the maximum memory usage in kilobytes (KB) of this instance of PrimeTime.

The memory usage of a multiprocess application, such as PrimeTime, reflects aggregate memory used across all its processes. Conceptually, this is the total amount of memory pages used for the entire application instance, without double counting shared pages (such as shared libraries and executables) across the application instance.

The memory usage of distributed processes is not included. For remote PrimeTime process memory usage, use the **report_host_usage** command.

EXAMPLES

The following example shows the memory usage information.

```
pt_shell> mem
8092
```

SEE ALSO

cputime(2)
report_host_usage(2)

merge_saif

Reads a list of SAIF files with their corresponding weights, annotates switching activity attributes with merged toggle_rate, glitch_rate, and merged static_probability for nets, pins, ports, and power arcs in the current instance, and generates a merged output SAIF file.

SYNTAX

```
status merge_saif
   -input_list saif_file_and_weight_list
   -strip_path inst_name
   [-path prefix]
   [-ignore ignore_name]
   [-exclude exclude_file_name]
   [-derate_glitch value]
   [-quiet]
   [-output merged_saif_name]
   [-simple_merge]
```

Data Types

```
saif_file_and_weight_list        list
inst_name                        string
prefix                           string
ignore_name                      string
exclude_file_name                string
value                            float
merged_saif_name                 string
```

ARGUMENTS

-input_list *saif_file_and_weight_list*

Specifies the name and the corresponding weight of the SAIF file list. *saif_file_and_weight_list* contains a list of { -input name.saif -weight number }, where "-input" and "-weight" are keywords. In addition, 0 < number < 100, and the sum of all weights is equal to 100. For example, { -input gate_back_1.saif -weight 20 -input gate_back_2.saif -weight 80 } means that the files gate_back_1.saif and gate_back_2.saif are weighted by 20% and 80%, respectively.

-strip_path *inst_name*

Specifies the name as it appears in each SAIF file of the current design instance you want to annotate. The **merge_saif** command annotates all subinstances in the hierarchy of the specified instance, as well as annotating the instance itself.

-path *prefix*

Specifies a relative path from the current design to the hierarchical low-level design for which the SAIF file has

been created. By default, absolute path names are used. Use this option if the SAIF file refers to an object in a hierarchy. Also if you want to read the switching information for a portion of the design, this option can be used. For example, if the SAIF was generated for the whole design, and you want to put switching information only on the nets/ports on the boundary and hierarchically under instance add14, then *-path* add14 option can be used.

-ignore *ignore_name*

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The **merge_saif** command ignores switching activity within the hierarchy of that instance in the SAIF file.

-exclude *exclude_file_name*

Specifies the name of a file that contains a list of names to be ignored. *exclude_file_name* is used when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the *-exclude* option. As with the *-ignore* option, the ignore names are recognized after the *-strip_path* argument.

-derate_glitch *value*

Specifies a default derating factor value to be used for inertial glitches in the SAIF file. If the *-derate_glitch* option is not specified, a default derating factor of 0.5 is used.

-quiet

Specifies quiet mode and for instance suppresses warnings on design objects in the SAIF file that could not be annotated on the design.

-output *merged_saif_name*

Specifies the name of the output merged SAIF file.

-simple_merge

Indicates that all SAIF files are to annotate 100% of the nets, ports and pins of the current instance, and the **merge_saif** command is to perform a simple merge of SAIF file data (without propagating the switching activity of non-annotated objects). Use the **report_switching_activity** command to verify that each SAIF file annotates 100% of the current instance.

DESCRIPTION

The **merge_saif** command reads a list of SAIF files with weight, computes the merged toggle_rate, glitch_rate, and static_probability, and annotates the switching activity attributes toggle_rate, glitch_rate, and static_probability for nets, ports, pins, and power arcs of the current instance. In addition, the **merge_saif** command optionally generates a merged output SAIF file. To identify the instance (and corresponding subinstances) you want to annotate, use the *-strip_path* argument. To specify a default derating factor other than 0.5, use the *-derate_glitch* option.

EXAMPLES

The following example reads and merges weighted SAIF files, annotates switching activity for the current design, and generates an output SAIF file.

```
pt_shell> set SAIF_DIR /home/user/ptpx/saif
```

```
pt_shell> set SAIF1 ${SAIF_DIR}/gate_back1.saif
pt_shell> set SAIF2 ${SAIF_DIR}/gate_back2.saif
pt_shell> set SAIF3 ${SAIF_DIR}/gate_back3.saif
pt_shell> set SAIF4 ${SAIF_DIR}/gate_back4.saif
pt_shell> merge_saif -input_list \
             " -input $SAIF1 -weight 10 \
               -input $SAIF2 -weight 20 \
               -input $SAIF3 -weight 30 \
               -input $SAIF4 -weight 40 " \
               -strip_path E/UUT -output merged_saif.saif
```

SEE ALSO

get_switching_activity(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_rtl_to_gate_name(2)
set_switching_activity(2)
write_saif(2)

open_mw_cel

Opens and reads a Milkyway design CEL view into the session.

SYNTAX

```
status open_mw_cel
   cel_name
   -library library_name
   -version version
```

Data Types

```
string cel_name
string library_name
int version
```

ARGUMENTS

cel_name

Specifies the name of the Milkyway CEL design to be read into the session.

-library *library_name*

Specifies the Milkyway design library in which the Milkyway CEL view exists.

-version *version*

Specifies which version of the Milkyway CEL view to be opened. By default, the command opens the latest version.

DESCRIPTION

The command opens the referenced Milkyway CEL view and builds up a linked design, including the internal contents of all soft macro blocks. Note that only one Milkyway CEL view can be opened at a time. To open another Milkyway CEL view, close the current one with the **close_mw_cel** command.

Soft macros are also flattened when being read into the design. The session **search_path** and **link_path** variables should be set before executing the **open_mw_cel** command.

EXAMPLES

The following example loads a Milkyway design CEL view.

```
pr_shell> open_mw_cel -library MyLibrary MyDesign
```

SEE ALSO

```
get_pg_pins(2)
set_starrc_options(2)
get_supply_nets(2)
link_path(3)
rail_sidefile_path(3)
search_path(3)
```

open_mw_lib

Opens a Milkyway library.

SYNTAX

```
collection open_mw_lib
   [-readonly | -write_ref]
   mw_lib
```

Data Types

```
mw_lib     string
```

ARGUMENTS

-readonly

Opens the Milkyway library only for reading. You cannot modify and save the library in this mode.

The **-readonly** and **-write_ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

-write_ref

Opens the reference library for writing. This option implies that the main library is opened for writing.

The **-readonly** and **-write-ref** options are mutually exclusive. The default is to open the main library as read/write and all reference libraries as read only.

mw_lib

Opens the Milkyway library.

DESCRIPTION

This command opens a Milkyway library. The two access permissions for an opened library are read only or read/write. Specifying the write permission implies that read permission is granted. If you specify the **-readonly** option, the command opens the main library and all reference libraries as read only. If you specify the **-write_ref** option, it opens the main library and all reference libraries as read/write.

If you open the library as read only, you can read from the library but you cannot write to the library.

Specifically, opening the Milkyway CEL view to write in it is not permitted. If you open the library as read/write, you

can modify it.

The opened Milkyway library is automatically set to be the current Milkyway library.

You cannot open more than one main library at the same time. To open another main library, you must first close the previous main library.

This command returns a collection of Milkyway libraries if it succeeds.

See the man pages for the **set_mw_lib_reference** and **set_mw_technology_file** commands for information on setting or changing reference libraries and technology files.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens a Milkyway library named *access05* for writing in the current session:

```
prompt> open_mw_lib access05
{"access05"}
```

SEE ALSO

```
close_mw_lib(2)
copy_mw_lib(2)
create_mw_lib(2)
current_mw_lib(2)
open_mw_lib(2)
rebuild_mw_lib(2)
rename_mw_lib(2)
report_mw_lib(2)
set_mw_lib_reference(2)
set_mw_technology_file(2)
write_mw_lib_files(2)
```

parse_proc_arguments

Parses the arguments passed into a Tcl procedure.

SYNTAX

```
string parse_proc_arguments -args arg_list
result_array

list arg_list
string result_array
```

ARGUMENTS

-args *arg_list*

Specified the list of arguments passed in to the Tcl procedure.

result_array

Specifies the name of the array into which the parsed arguments should be stored.

DESCRIPTION

The **parse_proc_arguments** command is used within a Tcl procedure to enable use of the -help option, and to support argument validation. It should typically be the first command called within a procedure. Procedures that use **parse_proc_arguments** will validate the semantics of the procedure arguments and generate the same syntax and semantic error messages as any application command (see the examples that follow).

When a procedure that uses **parse_proc_arguments** is invoked with the -help option, **parse_proc_arguments** will print help information (in the same style as using **help -verbose**) and will then cause the calling procedure to return. Similarly, if there was any type of error with the arguments (missing required arguments, invalid value, and so on), **parse_proc_arguments** will return a Tcl error and the calling procedure will terminate and return.

If you didn't specify -help, and the specified arguments were valid, the array variable *result_array* will contain each of the argument values, subscripted with the argument name. Note that the argument name here is NOT the names of the arguments in the procedure definition, but rather the names of the arguments as defined using the **define_proc_attributes** command.

The **parse_proc_arguments** command cannot be used outside of a procedure.

EXAMPLES

The following procedure shows how **parse_proc_arguments** is typically used. The argHandler procedure parses the arguments it receives. If the parse is successful, argHandler prints the options or values actually received.

```
proc argHandler args {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo "  $argname = $results($argname)"
  }
}
define_proc_attributes argHandler -info "argument processor" \
  -define_args \
  {{-Oos "oos help" AnOos one_of_string {required value_help {values {a b}}}}
   {-Int "int help" AnInt int optional}
   {-Float "float help" AFloat float optional}
   {-Bool "bool help" "" boolean optional}
   {-String "string help" AString string optional}
   {-List "list help" AList list optional}}
  {-IDup int dup AIDup int {optional merge_duplicates}}}
```

Invoking argHandler with the -help option generates the following:

```
prompt> argHandler -help
Usage: argHandler    # argument processor
        -Oos AnOos             (oos help:
                                Values: a, b)
        [-Int AnInt]           (int help)
        [-Float AFloat]        (float help)
        [-Bool]                (bool help)
        [-String AString]      (string help)
        [-List AList]          (list help)
```

Invoking argHandler with an invalid option causes the following output (and a Tcl error):

```
prompt> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer' (CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)
```

Invoking argHandler with valid arguments generates the following:

```
prompt> argHandler -Int 6 -Oos a
  -Oos = a
  -Int = 6
```

SEE ALSO

define_proc_attributes(2)
help(2)
proc(2)

print_message_info

Prints information about diagnostic messages that have occurred or have been limited.

SYNTAX

```
string print_message_info
   [-ids id_list]
   [-summary]
```

Data Types

```
id_list              list
```

ARGUMENTS

-ids *id_list*

List of message identifiers to report. Each entry can be a specific message or a glob-style pattern that matches one or more messages. If this option is omitted and no other options are given, all of the messages that have occurred or have been limited are reported.

-summary

Generate a summary of error, warning, and informational messages that have occurred so far.

DESCRIPTION

The **print_message_info** command enables you to print summary information about error, warning, and informational messages that have occurred or have been limited with the **set_message_info** command. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the **get_message_info** command, but you need to know the specific message ID. By default, the **print_message_info** command summarizes all of the information. It provides a single line for each message that has occurred or has been limited, and one summary line that shows the total number of errors, warnings, and informational messages that have occurred so far. If an *id_list* is given, only messages matching those pat- terns are displayed. If the *-summary* option is given, a summary is displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this does not show all messages with that prefix. Only those that have occurred or have been limited are displayed.

By default, the limit column shows the limit set by the **set_message_info** command. However, if the limit is not set and this type of message is included in the **sh_limited_messages** variable, this column shows the default limit of the **sh_message_limit** variable, and a symbol '*' also appears to show that this limit is from the **sh_message_limit** variable. Similarly, if the message is limited by **sh_global_per_message_limit**, this will be indicated by the symbol '-' following the limit column.

EXAMPLES

The following example uses the **print_message_info** command to display a few specific messages.

```
shell> print_message_info -ids [list "CMD*" APP-99]

  Id             Severity         Limit    Occurrences   Suppressed
  ---------------------------------------------------------------------
  CMD-005           Error         10000 -           10            3
  APP-99      Information             1             0            0
```

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following example shows how to use the **print_message_info** command in this way. The symbol '*' in the limit column of PARA-004 means that this message is in the **sh_limited_messages** variable and this limit is the **sh_message_limit** variable.

```
shell> print_message_info

 Id             Severity         Limit    Occurrences   Suppressed
 ---------------------------------------------------------------------
 CMD-005           Error         10000 -           12            0
 PARA-004        Warning           100 *            1            0
 APP-027     Information           100            150           50
 APP-99      Information         10000 -            1            0

 Diagnostics summary: 12 errors, 151 warnings, 1 informational
```

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with the **set_message_info** command, or the limit is from the **sh_message_limit** variable.

The sorting is by Severity (Error, Warning, Information), then by Occurrences, in descending order, then by Id.

SEE ALSO

```
get_message_info(2)
set_message_info(2)
suppress_message(2)
sh_global_per_message_limit(3)
sh_message_limit(3)
sh_limited_messages(3)
```

print_suppressed_messages

Displays an alphabetical list of message IDs that are currently suppressed.

SYNTAX

```
string print_suppressed_messages
```

ARGUMENTS

The **print_suppressed_messages** command has no arguments.

DESCRIPTION

The **print_suppressed_messages** command displays all messages that you suppressed using the **suppress_message** command. The messages are listed in alphabetical order. You only can suppress informational and warning messages. The result of **print_suppressed_messages** is always the empty string.

EXAMPLES

The following example shows the output from the **print_suppressed_messages** command:

```
prompt> print_suppressed_messages
No messages are suppressed

prompt> suppress_message {XYZ-001 CMD-029 UI-1}

prompt> print_suppressed_messages
The following 3 messages are suppressed:
        CMD-029, UI-1, XYZ-001
```

SEE ALSO

suppress_message(2)
unsuppress_message(2)

printenv

Prints the value of environment variables.

SYNTAX

```
string printenv
   [variable_name]
```

Data Types

variable_name	string

ARGUMENTS

variable_name

Specifies the name of a single environment variable to print.

DESCRIPTION

The **printenv** command prints the values of the environment variables inherited from the parent process or set from within the application with the **setenv** command. When no arguments are specified, all environment variables are listed. When a single *variable_name* is specified, if that variable exists, its value is printed. There is no useful return value from **printenv**.

To retrieve the value of a single environment variable, use the **getenv** command.

EXAMPLES

The following examples show the output from the **printenv** command:

```
prompt> printenv SHELL
/bin/csh

prompt> printenv EDITOR
emacs
```

SEE ALSO

```
getenv(2)
setenv(2)
```

printvar

Prints the values of one or more variables.

SYNTAX

```
string printvar
   [pattern]
   [-user_defined | -application]
```

Data Types

```
pattern      string
```

ARGUMENTS

pattern

Prints variable names that match *pattern*. The optional *pattern* argument can include the wildcard characters * (asterisk) and ? (question mark). If not specified, all variables are printed.

-user_defined

Shows only user-defined variables. This option is mutually exclusive with the **-application** option.

-application

Shows only application variables. This option is mutually exclusive with the **-user_defined** option.

DESCRIPTION

The **printvar** command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of application and user-defined variables. The **-user_defined** option limits the variables to those that you have defined. The **-application** option limits the variables to those created by the application. The two options are mutually exclusive and cannot be used together.

Some variables are suppressed from the output of **printvar**. For example, the Tcl environment variable array *env* is not shown. To see the environment variables, use the **printenv** command. Also, the Tcl variable **errorInfo** is not shown. To see the error stack, use the **error_info** command.

EXAMPLES

The following command prints the values of all of the variables:

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern *sh*a**:

```
prompt> printvar -application sh*a*
sh_arch                 = "sparcOS5"
sh_command_abbrev_mode  = "Anywhere"
sh_enable_page_mode     = "false"
sh_new_variable_message = "false"
sh_new_variable_message_in_proc = "false"
sh_source_uses_search_path = "false"
```

The following command prints the value of the **search_path** variable:

```
prompt> printvar search_path
search_path             = ".  /designs/newcpu/v1.6  /lib/cmos"
```

The following command prints the values of the user-defined variables:

```
prompt> printvar -user_defined
a                     = "6"
designDir             = "/u/designs"
```

SEE ALSO

```
error_info(2)
printenv(2)
setenv(2)
```

proc_args

Displays the formal parameters of a procedure.

SYNTAX

```
string proc_args
   proc_name
```

Data Types

proc_name	string

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_args** command is used to display the names of the formal parameters of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *args* argument.

EXAMPLES

This example shows the output of **proc_args** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_args plus
a b

prompt> info args plus
a b

prompt>
```

SEE ALSO

```
info(2)
proc(2)
proc_body(2)
```

proc_body

Displays the body of a procedure.

SYNTAX

```
string proc_body
   proc_name
```

Data Types

proc_name	string

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_body** command is used to display the body (contents) of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *body* argument.

EXAMPLES

This example shows the output of **proc_body** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_body plus
 return [expr $a  + $ b]

prompt> info body plus
 return [expr $a  + $ b]

prompt>
```

SEE ALSO

```
info(2)
proc(2)
proc_args(2)
```

query_cell_instances

Queries the instances of a mapped cell within the active scope. This is UPF query command similar to the **get_cells -of_object** PrimeTime command.

SYNTAX

```
list query_cell_instances
   [-domain domain_name]
   ref_name
```

Data Types

```
domain_name        string
ref_name           string
```

ARGUMENTS

-domain *domain_name*

Limits the search to the instances in the power domain specified.

ref_name

Specifies a reference name (library cell or design) that is used to locate an instance.

DESCRIPTION

The **query_cell_instances** command queries the instances of a mapped cell within the active scope.

SEE ALSO

get_cells(2)

query_cell_mapped

Returns cells that are mapped to a specified instance.

SYNTAX

```
string query_cell_mapped
  instance_name
```

Data Types

instance_name	string

ARGUMENTS

instance_name

Specifies the instance to query.

DESCRIPTION

The **query_cell_mapped** command returns cells that are mapped to the specified instance. This UPF query command is similar to the **get_lib_cells -of_objects** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

get_lib_cells(2)

query_net_ports

Returns a list of ports that are logically connected to the specified net.

SYNTAX

```
collection query_net_ports
   [-transitive false | true]
   [-leaf]
   net_name
```

Data Types

```
net_name     string
```

ARGUMENTS

-transitive false | true

When you set the **-transitive** option to **true**, the **query_net_ports** command applies to the descendants of the elements. The default is **false**.

-leaf

Returns only leaf cell ports.

net_name

Specifies the net to be queried.

DESCRIPTION

The **query_net_ports** commands returns a list of ports that are logically connected to the specified net. If no ports are connected, a null string is returned. This is UPF query command.

By default, this command searches both nonleaf and leaf cell ports. To search only leaf cell ports, use the **-leaf** option.

SEE ALSO

```
get_ports(2)
```

query_objects

Searches for and displays objects in the database.

SYNTAX

```
string query_objects
   [-verbose]
   [-truncate elem_count]
   [-class class_name]
   object_spec
```

Data Types

class_name	string
elem_count	int
object_spec	list

ARGUMENTS

-verbose

Displays the class of each object found. By default, only the name of each object is listed. With this option, each object name is preceded by its class (see the EXAMPLES).

-truncate *elem_count*

Truncates display to *elem_count* elements. By default, up to 100 elements display. To see more or less elements, use this option. To see all elements, set the *elem_count* value to 0.

-class *class_name*

For elements in the *object_spec* that are not collections, this is the class used when searching the database for objects which match the element. Valid classes are application-specific.

object_spec

Provides a list of objects to find and display. Each element in the list is either a collection or a pattern which will match some objects in the database. Patterns are explicitly searched for in the database with class *class_name*.

DESCRIPTION

The **query_objects** command finds and displays objects in the application's runtime database. The command

does not have a meaningful return value; it displays the objects found and returns an empty string.

The *object_spec* is a list containing collections and/or object names. For elements of the *object_spec* that are collections, **query_objects** simply displays the contents of the collection.

For elements of the *object_spec* that are names (or contain wildcard patterns), the **query_objects** command searches the database for objects of the class specified by the *class_name* option. Note: The **query_objects** command does not have a predefined implicit order of classes for which searches are initiated. If you do not specify the *class_name* option, only those elements that are collections are displayed. Messages are displayed for the other elements (see EXAMPLES).

To control the number of elements displayed, use the *-truncate* option. If the display is truncated, you see the ellipsis (…) as the last element. Note that if the default truncation occurs, a message displays showing the total number of elements that would have displayed (see EXAMPLES).

The **query_objects** command displays the output in either a Legacy format or in a Tcl compatible format. The default output format varies by tool but you can control the format yourself by setting the application variable query_objects_format. For example to force Tcl compatible output use this command:

```
pt_shell> set_app_var query_objects_format Tcl
Tcl
```

See below the differences between Legacy and Tcl formats.

NOTE: The output from the **query_objects** command looks similar to the output from any command that creates a collection; however, the result of the **query_objects** command is always an empty string.

EXAMPLES

These following examples show the basic usage of the **query_objects** command.

```
pt_shell> query_objects [get_cells o*]
{"or1", "or2", "or3"}
pt_shell> query_objects -class cell U*
{"U1", "U2"}
pt_shell> query_objects -verbose -class cell \
?            \[list U* [get_nets n1]]
{"cell:U1", "cell:U2", "net:n1"}
```

If the output format is Tcl, then the output looks like this:

```
pt_shell> query_objects [get_cells o*]
{or1 or2 or3}
pt_shell> query_objects -class cell U*
{U1 U2}
pt_shell> query_objects -verbose -class cell \
?            \[list U* [get_nets n1]]
{{cell U1} {cell U2} {net n1}}
```

When you omit the **-class** option, only those elements of the *object_spec* that are collections generate output. The other elements generate error messages.

```
pt_shell> query_objects \[list U* [get_nets n1] n*]
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{"n1"}
```

When the output is truncated, you get the ellipsis at the end of the display. For the following example, assume the

default truncation is 5 (it is actually 100).

```
pt_shell> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
pt_shell> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

SEE ALSO

collections(2)
get_cells(2)
get_clocks(2)
get_designs(2)
get_generated_clocks(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_path_groups(2)
get_pins(2)
get_ports(2)
get_qtm_ports(2)
get_timing_paths(2)

query_port_net

Queries the net logically connected to a specified port and returns the net name.

SYNTAX

```
list query_port_net
   [-conn high | low]
   port_name
```

Data Types

```
port_name     string
```

ARGUMENTS

-conn high | low

Queries the **high** (the default) or **low** connection. You can use this option only if the specified port is not on a leaf cell.

port_name

Specifies the port name to query.

DESCRIPTION

The **query_port_net** commands returns the net logically connected to a port. If no net is connected, the command returns a null string.

This is UPF query command.

SEE ALSO

get_nets(2)

quit

Exits the shell.

SYNTAX

`string` **quit**

ARGUMENTS

The **quit** command has no arguments.

DESCRIPTION

The **quit** command exits from the application. It is basically a synonym for the **exit** command with no arguments.

EXAMPLES

The following example exits the current session:

```
prompt> quit
```

SEE ALSO

`exit(2)`

read_db

Reads in one or more design or library files in Synopsys database (db) format.

SYNTAX

```
string read_db
   [-netlist_only]
   [-library]
   file_names
```

Data Types

```
file_names      list
```

ARGUMENTS

-netlist_only

For designs only; ignored for libraries. Indicates that only the netlist is to be read; attributes are not to be read. This can greatly increase performance while reading designs.

-library

Indicates that the file is a library db file; using this option optimizes the reading of large library db files.

file_names

Specifies names of one or more files to be read. Typically, only one file is read.

DESCRIPTION

Reads design and library information from Synopsys database (db) files into PrimeTime.

To locate files that have relative pathnames, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path** variable. Files that have absolute pathnames are loaded as though there were no **search_path** variable. To determine the file that the **read_db** command loads, use the **which** command.

Each file can contain one or more design objects. After the files are loaded, to view the design objects, use the **list_designs** and **list_libs** commands. The db file itself is used only during the read process, and is transformed into the internal format of PrimeTime during the **read_db** command . To remove designs and libraries, use the **remove_design** and **remove_lib** commands.

Synopsys design db files are used for many tools and can contain many complex attributes. However, only certain attributes apply to PrimeTime (for example, clocks, back-annotation). By default, these attributes are loaded from the db file. However, all information can also be brought into PrimeTime through scripts or other file formats (for example, SDF). In some cases (most notably SDF), you can save CPU time and memory usage by reading the information from the alternate source. In these cases, use the *-netlist_only* option to read the db file. The *-netlist_only* option speeds up the read process and drastically reduces memory usage in the case of SDF back-annotation. You can then read the SDF file using the **read_sdf** command.

If the **power_enable_analysis** variable is set to true, the **read_db** command also loads the power models in library db files.

EXAMPLES

In the following example, the file newcpu.db is read based on the *search_path* variable.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos

pt_shell> read_db newcpu.db
Loading db file '/designs/newcpu/v1.6/dbs/newcpu.db'
1
```

SEE ALSO

```
list_designs(2)
list_libs(2)
read_sdf(2)
remove_design(2)
remove_lib(2)
which(2)
search_path(3)
```

read_def

Annotates the design with data from a file in Design Exchange Format (DEF).

SYNTAX

```
status read_def
   .prod icc
   [-check_only]
   [-enforce_scaling]
   [-no_incremental]
   [-verbose]
   [-turn_via_to_inst]
   [-inexactly_matched_via_to_inst]
   [-lef lef_file_names]
   [-snet_no_shape_as_user_enter]
   [-snet_no_shape_as_detail_route]
   [-preserve_wire_ends]
   [-section_list]
   def_file_names
```

Data Types

```
. prod icc
lef_file_names                list
def_file_names                list
```

ARGUMENTS

-check_only

Analyzes the input DEF file without annotating any of the data on the design. It provides diagnostic information on the correctness and integrity of DEF file.

-enforce_scaling

Instructs the tool to continue to annotate the design with information given in the DEF file, even when conversion of data from DEF units to library units could result in roundoff errors.

-no_incremental

Removes all pre-existing physical annotations before reading in the physical information in the DEF file.

-verbose

Generates additional debugging messages of two types: information and warnings.

-turn_via_to_inst

Creates turn vias as via cell instances rather than path objects. Turn vias are represented in DEF as a single rectangle (usually a piece of dangling metal).

-inexactly_matched_via_to_inst

Creates inexactly matched vias as via cell instances. Inexactly matched vias are vias that match a Milkyway technology contact code's cut dimensions, but the enclosure dimensions are larger than those of the contact code. By default, the tool creates inexactly matched vias as contact arrays and extra paths on the enclosure layers.

-lef *lef_file_names*

Imports the rotated vias and design-specific nondefault rules from the LEF files into the design. The incremental LEF files are generated by the **write_def** command.

-snet_no_shape_as_user_enter

Marks the wire segments without "+ SHAPE" statement in SPECIALNETS with the route type "user enter".

-snet_no_shape_as_detail_route

Marks the wire segments without "+ SHAPE" statement in SPECIALNETS with the route type "detail route".

-preserve_wire_ends

Creates special power/ground wiring as paths without extensions (that is, square ends), special fill wiring as rectangles, and all other special and regular wiring as wires with half-width extensions (that is, square ends by half-width).

-section_list

Reads the specified sections (vias, components, nets, special nets, etc). Make sure that the specified sections are self-contained prior to using this option.

def_file_names

Specifies the name of the file from which to draw physical data to annotate onto the design. This file must be in Design Exchange Format (DEF). For multiple files, the DEF files are read in incrementally, and you must specify the DEF files in the correct order. It strongly recommended that you remove overlapping floorplanning data, such as ROWS, and duplicate routing geometry data in the DEF files. Otherwise, these conditions might be error-prone in the subsequent flow.

DESCRIPTION

This command annotates the design with physical design data taken from a file in Design Exchange Format (DEF). The command can read both uncompressed and compressed (in gzip format) DEF files.

The command reads physical data associated with the current design from a DEF file. If cell, port, or net specified in the DEF file does not exist in the database, the corresponding object is created in the Milkyway database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the design is annotated with the physical data defined in the DEF file named top.def.

```
prompt> read_def top.def
```

SEE ALSO

write_def(2)

read_em_rules

Reads electromigration (EM) rules from the current Milkyway library or from a file in a supported format.

SYNTAX

```
status read_em_rules
   -format rule_format
   [rule_file]
   [-design2itf_map map_file]
```

Data Types

```
string rule_format
string rule_file
string map_file
```

ARGUMENTS

-format *rule_format*

Specifies the format of the electromigration rules. The allowed values are alf, mw_tech_file, mw_cached_em_rule, and tluplus. If *rule_format* is alf, an ALF rule file must be provided. If *rule_format* is tluplus, a TLUPlus file and a layer name mapping file must be provided.

rule_file

Specifies the name of the file that contains the electromigration rules.

-design2itf_map *map_file*

Specifies the name of the mapping file that defines the name mapping between the layer names used in design database and those in the TLUPlus rule file. This option is required only when the -format option is set to tluplus.

DESCRIPTION

The command reads the specified electromigration (EM) rules into the analysis session. When successful, the new rules replace the existing ones. These electromigration rules are used to perform electromigration analysis on the power supply network. See the **report_em_violations** man page for further details.

This command currently supports four different formats: Milkyway-cached EM rules (mw_cached_em_rule), technology file (mw_tech_file), ALF (alf), and TLUPLUS (tluplus).

When the -format option is specified as mw_cached_em_rule, the electromigration rules are read from the Milkyway cached EM rules. No *rule_file* is needed.

When the -format option is specified as mw_tech_file, the electromigration rules are read from the technology file in the Milkyway library. No *rule_file* is needed.

When the -format option is specified as alf, an ALF rule file must be specified. Only the electromigration rules in this ALF file are read and all other content is ignored.

When the -format option is specified as tluplus, a TLUPlus rule file is required. Only the electromigration rules in the TLUPlus file are read and all other content is ignored. You must also provide a mapping file which defines the mapping between the layers in the design database and those in the TLUPlus file with the -design2itf_map option.

In this mapping file, the conducting_layers section defines the layer name mapping for metal layers. The via_layers section defines the layer name mapping for via layers. In each section, the layer names used in design database are listed in the first column, where the layer names used in the TLUPlus file are in the second column. Lines starting with "*" are considered as comments.

When the **open_mw_cel** command is run, the Milkyway-cached electromigration rules are read by default. If there are no rules stored in the Milkyway library, the electromigration rules stored in the technology file are read instead.

When electromigration rules are specified in an ALF file, the file might contain library information in addition to the electromigration rules. This command performs only limited syntax checking of the non-electromigration portions of the electromigration rules file and does not store this information. The ALF file may have multiple electromigration rules specified, for example, static, average, absolute_average, rms and peak. During static EM analysis, only the averaged rule is used. During dynamic electromigration analysis, only average, rms and peak rules are used, and users can specify one or all of the three to be reported. See more details in the **report_em_violations** man page.

When electromigration rules are specified in a TLUPlus file, this command parsed in the electromigration rule related information specified at header comment section of the TLUPlus file. This electromigration rule portion starts with string "#EM_INFORMATION", ends with string "#END_OF_EM_INFORMATION". The TLUPlus file might have multiple electromigration rules specified, for example, average, rms and peak. During static electromigration analysis, only average rule is used. During dynamic EM analysis, only average, rms and peak rules are used, and users can specify one or all of the three to be reported. See more details in the **report_em_violations** man page.

Regardless of the electromigration rule input format, the electromigration rules establish the current density limits for the layout geometries on a device layer.

EXAMPLES

This example replaces the current electromigration rules with the rules read from *MyFile.alf*:

```
pr_shell> read_em_rules -format alf MyFile.alf
1
```

This example replaces the current electromigration rules with the rules from Milkyway cached EM rules.

```
pr_shell> read_em_rules -format mw_cached_em_rule
1
```

This example replaces the current electromigration rules with the rules from Milkyway technology file.

```
pr_shell> read_em_rules -format mw_tech_file
1
```

This example replaces the current electromigration rules with the rules read from *MyFile.tluplus*:

```
pr_shell> read_em_rules -format tluplus MyFile.tluplus -design2itf_map layer_mapping_file
1
```

SEE ALSO

```
calculate_rail_voltage(2)
report_em_rules(2)
report_em_violations(2)
open_mw_cel(2)
```

read_file

Reads a netlist or library file. This is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
status read_file
  [-format db | edif | vhdl | verilog]
  [-single_file ns1]
  [-names_file ns2]
  [-define ns3]
  file_list
```

Data Types

```
ns1            string
ns2            string
ns3            string
file_list      list
```

ARGUMENTS

-format db | edif | vhdl | verilog

Specifies the file format.

-single_file *ns1*

Not supported by PrimeTime.

-names_file *ns2*

Not supported by PrimeTime.

-define *ns3*

Not supported by PrimeTime.

file_list

Specifies a list of files that are to be read.

DESCRIPTION

The **read_file** command reads in one or more netlist or library files. The command exists in PrimeTime for

compatibility with Design Compiler. Complete information about the **read_file** command can be found in the Design Compiler documentation. The supported method for reading netlist or library files is to use the following commands: **read_db**, **read_verilog**, **read_edif**, and **read_vhdl**.

SEE ALSO

```
read_db(2)
read_verilog(2)
read_vhdl(2)
```

read_lef

Create a Milkyway reference library ready for place and route from input LEF files.

SYNTAX

```
int read_lef
   [ -lib_name <lib_name> ]
   [ -tech_lef_files <tech_lef_files> ]
   [ -cell_lef_files <cell_lef_files> ]
   [ -layer_mapping <layer_mapping_file_name> ]
   [ -overwrite_existing_tech ]
   [ -ignore_cell_geom ]
   [ -advanced_lib_prep_mode ]
   [ -cell_version <merge | overwrite | new | ignore> ]
   [ -cell_boundary by_cell_size | by_overlap_layer ]
   [ -overlap_in_block_ring_only ]

string lib_name
string tech_lef_files
string cell_lef_files
string layer_mapping_file_name
```

ARGUMENTS

-lib_name *<lib_name>*

Specifies the Milkyway library in which the library information and reference cells will be created. The string can include the path of the library.

If no path is specified, the library is resolved from the current working directory. The read_lef command creates a new reference library based on LEF information if you do not have an existing library.

-tech_lef_files *<tech_lef_files>*

Specifies the Tech input LEF files. You can also specify multiple LEF files if, for example, you have separate LEF files for technology information and antenna information.

-cell_lef_files *<cell_lef_files>*

Specifies the Cell LEF input files. You can also specify multiple LEF files if, for example, you have separate LEF files for standard cells and macros.

-layer_mapping *<layer_mapping_file_name>*

Specifies a layer mapping file that contains the LEF layer name and Milkyway layer number. If you do not specify this file, Milkyway follows the LEF layer order with the first layer in LEF mapping to layer 1 in Milkyway and the second layer in LEF mapping to layer 2 in Milkyway, and so on. Specifying a layer mapping file is recommended.

-overwrite_existing_tech

This option is not selected by default. When you select this option, all the existing technology information from the library is removed and the new technology information from the Tech LEF Files is used instead.

-ignore_cell_geom

This option is not selected by default. When you select this option, the pin, rectangle, polygon and via objects are not created. After LEF input, you must input the GDSII data to create the physical information. This option is typically used for the Tech LEF + Physical Cell GDS flow.

-advanced_lib_prep_mode

This option is not selected by default. In the default mode, read_lef automatically completes all library preparation steps and creates a Milkyway library ready for place and route.

If this option is selected, you must manually run the steps that follow Import LEF in order to create a Milkyway Library (steps such as Extract BPV, Set PR Boundary, Set Property for Multiple Height Cells, and Define Wire Track). Use this option for special cases or for advanced library preparation.

-cell_version <*merge* | *overwrite* | *new* | *ignore*>

Merge with Existing Cell: This option is selected by default. When selected, the latest cell version will include the information defined in the LEF file and the existing database information. Cell-related information such as macro pin and macro obstructions can be appended to the existing information.

Overwrite Existing Cell: This option is not selected by default. If this option is selected, the latest cell versions will be overwritten by the LEF macro cells.

Make New Cell Version: This option is not selected by default. If this option is selected, the new cell versions will be created by the macro cells defined in LEF. The old version will be maintained but a new version based on the LEF information will be created.

Ignore LEF Cell: This option is not selected by default. If this option is selected, macros defined in the LEF file with the same name will be ignored when you import the LEF file.

-cell_boundary by_cell_size | by_overlap_layer

Specifies whether to extract rectangular or rectilinear cell boundaries for macro cells. With the **by_cell_size** option setting (the default behavior), a rectangular cell boundary is derived from the SIZE parameter in the LEF file. With the **by_overlap_layer** option setting, a rectilinear cell boundary (possibly non-rectangular) is derived from the OVERLAP layer in the LEF file.

-overlap_in_block_ring_only

Specifies whether the **-cell_boundary by_overlap_layer** setting, which selects rectilinear boundary extraction, applies to all macro cells or only BLOCK and RING type macro cells. The default is to perform rectilinear boundary extraction on all macros. With the **-overlap_in_block_ring_only** option, rectilinear boundary extraction is performed only on BLOCK and RING type macro cells. The **-cell_boundary by_overlap_layer** option has an effect only when **-cell_boundary by_overlap_layer** is used.

DESCRIPTION

The **read_lef** command create a Milkyway reference library ready for place and route from input LEF files.

EXAMPLES

The following example read LEF files to create a milkyway library "*design*".

```
read_lef -lib_name design -tech_lef_files tech.lef -cell_lef_files cell.lef
```

SEE ALSO

auNLOApi(2)
write_lef(2)

read_lib

Reads in a Synopsys library (.lib) file.

SYNTAX

```
status read_lib
   file_name
```

Data Types

file_name	list

ARGUMENTS

file_name

Specifies the name of a library file to read.

DESCRIPTION

The **read_lib** command reads a Synopsys library (.lib) file into PrimeTime using the PrimeTime external reader (ptxr). To locate files that have relative path names, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path** variable. Files that have absolute path names are loaded as though there was no **search_path**. To determine the file that the **read_lib** command loads, use the **which** command. After the library file is loaded, to list the library objects, use the **list_libs** command. To remove libraries, use the **remove_lib** command.

If the **power_enable_analysis** variable is set to **true**, the **read_lib** command also loads power models. If the variable is **false**, neither the **read_lib** nor **link_design** command nor the first power related command loads power models. The incremental power model loading only works on db files by using either the **link_library** or **read_db** command or both of these commands. In other words, if you use .lib files and you want to do power analysis, you must set the **power_enable_analysis** variable to **true** and use the **read_lib** command to read in your .lib files.

For more information, see the Library Compiler documentation.

EXAMPLES

In the following example, the file bus1.lib is read based on the **search_path**.

```
pt_shell> set search_path "/designs/newcpu/v1.6 /libs/cmos"
/designs/newcpu/v1.6 /libs/cmos

pt_shell> read_lib bus1.lib
Beginning read_lib...
Reading '/libs/cmos/bus1.lib' ...
Technology library 'bus1' read successfully
1
pt_shell> list_libs

Library Registry:
    bus1           bus1.lib:bus1

1
```

SEE ALSO

```
list_libs(2)
remove_lib(2)
which(2)
power_enable_analysis(3)
search_path(3)
```

read_parasitics

Reads net parasitics information from an SPEF, DSPF, RSPF, PARA, or Galaxy Parasitics Database (GPD) and uses it to annotate the currently linked design.

SYNTAX

```
status read_parasitics
   [-format file_fmt]
   [-complete_with completion_type]
   [-lumped_cap_only]
   [-pin_cap_included]
   [-path prefix]
   [-coupling_reduction_factor factor]
   [-triplet_type ttype]
   [-verbose]
   [-syntax_only]
   [-ilm_context]
   [-eco file_name]
   [-original_file_name file_name]
   [-keep_capacitive_coupling]
   [-x_offset xlist]
   [-y_offset ylist]
   [-axis_flip flip_list]
   [-rotation rot_list]
   file_names
```

Data Types

file_fmt	string
completion_type	string
prefix	string
factor	float
ttype	string
file_name	string
xlist	string
ylist	string
flip_list	string
rot_list	string
file_names	string

ARGUMENTS

-format *file_fmt*

Specifies the format of the parasitics file. The allowed values are SPEF, DSPF, RSPF, PARA (Milkyway), and GPD. If the **-format** option is not specified, the application can determine whether the file is GPD, SPEF, DSPF, RSPF, or a compressed version of those three ASCII formats. However, to read a file in PARA you must specify the **-format PARA** option.

-complete_with *completion_type*

This option does not apply to the RSPF format.

This option indicates that a net with partially annotated parasitics is to be completed by inserting capacitances and resistances according to the *completion_type* argument. The allowed values are **zero**, which completes the net by inserting zero capacitances and resistances, and **wlm**, which completes the net by inserting capacitances and resistances derived from wire load models.

This option is equivalent to reading the parasitics file and then using the **complete_net_parasitics -complete_with** command.

Note: The **complete_net_parasitics** and **read_parasitics -complete_with** commands complete a net only if all missing segments are between two pins and the nets are partially annotated (nets are not affected if they are fully annotated or have no annotation at all). The net must also be hierarchical, so that if the parasitics for the block-level parts of a net are missing, those parasitics could exist in the top-level net. If any of these conditions are not met, you must manually correct the SPEF or DSPF file.

-lumped_cap_only

This option does not apply to the GPD format.

This option indicates that only the total capacitance of nets is to be annotated as a lumped capacitance on the annotated nets. The RC networks specified in the parasitics file are discarded. The annotated lumped capacitance is the capacitance specified when the net is declared in the parasitics file.

-pin_cap_included

Indicates that the RC networks are to include the pin capacitances. By default, the RC network does not include pin capacitances. This option does not apply to the RSPF format. The RC pi model in RSPF format has to always include effect of pin capacitances.

-path *prefix*

Specifies a list of relative paths from the current design to the hierarchical instance names for which the parasitics file has been created. By default, absolute path names are used. Use this option if the parasitics file refers to an object (for example, *net*) in a hierarchy (for example, *hier*). Do not use this option if the parasitics file refers to an absolute path (for example, hier/net). If more than one prefix is supplied, all of the instances of the design can be annotated simultaneously.

-coupling_reduction_factor *factor*

This option applies only to the SPEF format format. A positive floating point number that specifies the factor to apply when reducing coupling capacitances to grounded capacitances. The default is 1.0. This option is disabled if the **-keep_capacitive_coupling** option is specified. The command fails if both options are specified.

-triplet_type *ttype*

This option applies only to the SPEF and PARA formats. Capacitor and resistor values in the SPEF and PARA files can be specified as triplets with minimum, typical, and maximum values. Be default, PrimeTime uses the maximum values. Use this option to specify which values are used by PrimeTime: **min**, **typ**, or **max**. The default is *max*.

-verbose

Indicates that the **report_annotated_parasitics** report is to be generated when the parasitic file has been read. By default, after reading the parasitics file, the **report_annotated_parasitics** command is not executed.

-syntax_only

Indicates that the **read_parasitics** command is to parse the file for syntax errors without performing any parasitic annotation. Use this option to troubleshoot your parasitics file and avoid generating error messages during the actual annotation. No design is required to use the **-syntax_only** option.

-ilm_context

Indicates that the annotation is being performed in the presence of Interface Logic Models (ILMs). An original design parasitics can be used to annotate a design with ILMs using this option. This option does not issue error messages for missing nets, cells, and pins.

-eco *file_name*

This option specifies the ECO (incremental) parasitic file to be read along with the original parasitic file. The ECO parasitic annotation will update the original annotations. Parasitic errors and warnings for the original parasitic file will be suppressed, and only errors and warnings related to the ECO file will be issued. User should refer to the initial **read_parasitics** command to check for all warnings and errors. Note that in HyperScale top, this command cannot be used to update the parasitic annotations of a block.

This option is presently also supporting the old behavior which indicates that the file being currently annotated is an ECO parasitics from StarRC. PrimeTime SI can read ECO parasitics that are written out only by StarRC. The ECO parasitics can be annotated only when there are some existing parasitics that are already annotated. ECO parasitic files contain re-extracted parasitics for only the ECO nets and their immediate coupling neighbors and do not contain all the nets of the design. Incremental analysis can be performed after reading ECO parasitics.

-original_file_name *file_name*

This option can only be used when the **-eco** option is being used. If the original annotation is performed through multiple parasitic files into PrimeTime SI, the ECO parasitic file corresponds to one of the original files (because it corresponds to one extracted database in StarRC). PrimeTime SI attempts to determine the corresponding original file, but it is not always possible. You can use this option to specify which original parasitic file the ECO file correspond to.

-keep_capacitive_coupling

Indicates that the cross capacitors are to be kept in the RC networks data structure. This facilitates the capacitive crosstalk analysis, but does not turn it on. This option disables the **-coupling_reduction_factor** option. The command fails if both options are specified. All coupling capacitors are split to ground with a factor of 1.0 if crosstalk analysis is not activated. This option applies to both the SPEF and the GPD format and requires a PrimeTime SI license.

-x_offset *xlist*

Indicates a list of x-offsets for all instances specified in the **-path** option. The order in which each x-offsets appears in the **-x_offset** option must correspond to the order of each instance in the **-path** option. Specify the x-offsets in nanometers.

-y_offset *ylist*

Indicates a list of y-offsets for all instances specified in the **-path** option. The order in which each y-offset appears in the **-y_offset** option must correspond to the order of each instance in the **-path** option. Specify the y-offsets in nanometers.

-axis_flip *flip_list*

Indicates a list of axis flips for all instances specified in the **-path** option. The order in which each axis flip appears in the **-axis_flip** option must correspond to the order of each instance in the **-path** option. The allowed values are *flip_x*, which indicates a flip across the x-axis, *flip_y*, which indicates a flip across the y-axis, *flip_both*, which indicates a flip along both axis and *flip_none*, which indicates no flip.

-rotation *rot_list*

Indicates a list of rotations for all instances specified in the **-path** option. The order in which each rotation appears in the **-rotation** option must correspond to the order of each instance in the **-path** option. The allowed values are **rotate_none**, **rotate_90**, **rotate_180**, and **rotate_270**, which indicate a clockwise rotation of 0, 90, 180, and 270 degrees, respectively. The default is **rotate_none**.

file_names

When the format is SPEF, DSPF, RSPF, or GPD, this option specifies a list of files from which parasitics information is to be read.

DESCRIPTION

The **read_parasitics** command reads parasitics information from a disk file and annotates it on nets of the current design. This command supports the SPEF, DSPF, RSPF, and Galaxy Parasitics Database (GPD) formats. For SPEF, RSPF, and DSPF, the file can be a simple ASCII file, or it can be compressed with gzip. The **read_parasitics** command automatically detects this format information from the file, but you can specify the base format using the **-format** option.

GPD is an efficient format for sharing parasitics among Synopsys applications. You can use the PrimeTime **write_parasitics** command to write a parasitics file in GPD.

You can specify parasitics in either reduced or detailed form. Reduced parasitics consist of an RC pi model specified for each driver pin of a net. An RC pi model contains two capacitances and one resistance. The first capacitance connects to the net driver pin; the resistance connects to the net driver pin and to a subnode to which the second capacitance connects. Both capacitances connect to the ground. The parasitics file, using the reduced form, also specifies the pin-to-pin net delays. The RC pi model is used to compute the effective capacitance of the nets at each driver of the net. The effective capacitance determines the cell delays and output slews. You can specify reduced parasitics with the SPEF, DSPF, or RSPF format.

Detailed parasitics consist of a network of resistances and capacitances for each net specified in the parasitics file. The capacitances must be with respect to the ground. Coupling capacitances are not supported unless the **-keep_capacitive_coupling** option is specified and you are using the SPEF or GPD format. Resistances connected to the ground are not connected. The detailed RC network must be complete; incomplete RC networks are discarded but the delays are computed using the incomplete subset of resistances and capacitances. You can specify detailed parasitics with the SPEF, GPD, DSPF, or RSPF formats.

The **read_parasitics** command does not overwrite lumped parasitics (set using the **set_load** or **set_resistance** command) if they already exist on the design. Instead, a warning is issued. If the lumped parasitics are subsequently removed (using the **remove_capacitance** and **remove_resistance** commands) PrimeTime reverts to the reduced or detailed parasitics.

Incremental annotations are supported; if a small number of nets are affected, a full timing update is not required if the design is already timed. The maximum number of affected nets that avoid a full timing update depends on the total design size.

Multiple resistances and capacitances between the same nodes are also accepted. The values of multiple capacitances between a node are added together and multiple resistances between the same two nodes are kept separately and are considered during the delay calculation.

It is most efficient to read the top level last to minimize the amount of stitching that must be performed.

Net and instance pin names in the design must match instance names in the parasitics file. For example, if you create the parasitics file from a design using VHDL naming conventions, the design name must use VHDL naming conventions.

Parasitics of HyperScale instances are automatically read by PrimeTime when these instances are instantiated at top level, and user-specified parasitics reading of these instances is automatically skipped. Note that if the **-path** option in **read_parasitics** includes multiple instances among which some are HyperScale instances and the others are not, the parasitics reading for those non-HyperScale instances are not skipped.

To remove parasitics that were read and annotated with the **read_parasitics** command, use the **remove_annotated_parasitics** command. The **reset_design** command removes all attributes from the current design, including annotated parasitics.

In PrimeTime, the delay calculation algorithms impose no limit on the number of resistances and capacitances that can be annotated onto a single network; however, there are two shell variables you can manipulate to track and filter annotations that might cause unexpectedly large runtimes. Use the **parasitics_warning_net_size** variable to specify the threshold number of nodes for which the PARA-003 message is to be issued, indicating that a large amount of annotation has been encountered for a given network. Similarly, use the **parasitics_rejection_net_size** variable to specify when such detailed annotation is to be rejected in favor of lumped annotation, issuing the PARA-004 warning message.

PrimeTime can also load locations information from parasitic files. This capability is controlled by the **read_parasitics_load_locations** variable. By default, this variable is **false**, and no locations are read into PrimeTime. You can set this variable to *true* to load locations.

Hierarchical parasitics with locations require geometric transformations to be associated with the sub-blocks (i.e.; -path option). PrimeTime computes these transformations based on: user defined transformation; StarRc location guidance for SPEF/GPD; Automatic determination of location transformation.

When reading PARA format the parasitics might not be the same as what is specified by the **-triplet_type** option. This happens when the specified type does not exist. In that case, the **read_parasitics** command chooses something according to the following table:

```
                            Parasitics stored as
   triplet_type      |   typ   min-max    min-max-typ
 --------------------+---------------------------------
     min             |   typ     min          min
     typ             |   typ     max          typ
     max             |   typ     max          max
```

PrimeTime can currently read multicorner parasitic file and select one corner of interest. You must specify the corner name by using the **parasitic_corner_name** variable, before using the **read_parasitics** command. After the parasitic corner is set, it cannot be changed between subsequent **read_parasitics** commands. Also, reading a multicorner parasitic file without setting this variable results in an error. With this feature, the parasitics for only one specific corner from the set of corners is read and annotated. All other parasitic commands would operate on this given corner. The feature is supported for both SPEF and GPD.

EXAMPLES

The following example reads the adder.spef parasitics file from the disk and uses it to annotate the RC parasitics on the current design. The parasitics file contains a description of the RC network for nets of a design.

```
pt_shell> read_parasitics adder.spef
```

The following example shows annotating hierarchical parasitics files. The second command reads parasitics information, for instance u1 of design mult16 from the disk file mult16_u1.spef. The parasitics are annotated on the design, MY_DESIGN. The **-verbose** option enables the automatic call to the **report_annotated_parasitics** command, which would report RC networks of the boundary nets of instance u1 that are not complete.

Pin capacitances are included in the annotated parasitics, as specified by the **-pin_cap_included** option. The

second command reads another parasitics file in incremental mode, so that parasitics annotated on the boundary nets of instance u1 are not discarded.

```
pt_shell> current_design MY_DESIGN
pt_shell> read_parasitics -pin_cap_included -path u1 mult16_u1.spef
pt_shell> read_parasitics -pin_cap_included -verbose mydesign.spef
```

The following example shows hierarchical back annotation. The top_level design contains two instances, inst0 and inst1, of the mydesign design.

```
pt_shell> read_parasitics top_level.spf
pt_shell> read_parasitics mydesign.spf -path inst0
pt_shell> read_parasitics mydesign.spf -path inst1
```

The following example removes annotated parasitics information from the current design.

```
pt_shell> remove_annotated_parasitics
```

The following example loads locations into PrimeTime memory.

```
pt_shell> set read_parasitics_load_locations true
pt_shell> read_parasitics adder.spef
```

The following example reads the C_2 corner from a multicorner parasitic file.

```
pt_shell> set parasitic_corner_name "C_2"
pt_shell> read_parasitics multi_corner.spef
```

SEE ALSO

```
complete_net_parasitics(2)
remove_annotated_parasitics(2)
report_annotated_parasitics(2)
reset_design(2)
write_parasitics(2)
read_parasitics_load_locations(3)
si_enable_analysis(3)
si_filter_per_aggr_noise_peak_ratio(3)
PARA-003(n)
PARA-004(n)
```

read_saif

Reads a Switching Activity Interchange Format (SAIF) file and annotates switching activity information on nets, pins, ports, and cells in the current design.

SYNTAX

```
status read_saif
   file_name
   [-strip_path prefix]
   [-report_inconsistent_annotation report_file]
   [-path prefix]
   [-ignore ignore_name]
   [-exclude exclude_file_name]
   [-derate_glitch value]
   [-quiet]
```

Data Types

```
file_name            string
prefix               string
report_file          string
ignore_name          string
exclude_file_name    string
value                float
```

ARGUMENTS

file_name

Specifies the name of the SAIF file to read. If the file name ends with .gz, .Z, or .bz2, it is read as a gzipped file, a compressed file, or a bzip2ed file, respectively. Otherwise, it is assumed that the file is uncompressed.

-strip_path *prefix*

The **read_saif** command assumes that the current design is instantiated as an instance in the testbench used to generate the SAIF file. The current design, therefore, appears as an instance in the SAIF file. The **-strip_path** argument specifies the name of the instance of the current design as it appears in the SAIF file.

The **read_saif** command annotates all subinstances in the hierarchy of the specified instance and annotates the instance itself. Enter each instance name completely, without any trailing hierarchy separator (/). For example, if the current design is instantiated in the simulation environment as TOP/DUT/U1, use -strip_path TOP/DUT/U1 and not -strip_path TOP/DUT/U.

-report_inconsistent_annotation *report_file*

This option specifies the report file name containing objects whose activity is overwritten with different values. When specified, read_saif checks the consistency of the annotated values on objects whose activity values are

overwritten, and outputs any inconconsistencies in the report_file. If none occur, the report file is empty.

-path ***prefix***

Specifies a relative path from the current design to the hierarchical low-level design for which the SAIF file has been created. By default, absolute path names are used. Use this option if the SAIF file refers to an object in a hierarchy. If you want to read the switching information for a portion of the design, you can also use this option. For example, if the SAIF was generated for the whole design and you want to put only switching information on the nets/ports on the boundary and hierarchically under instance add14, then use the **-path add14** option.

-ignore ***ignore_name***

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The **read_saif** command ignores switching activity within the hierarchy of that instance in the SAIF file. *ignore_name* can be a hierarchical name separated by a slash (/). The **-strip_path** option is applied first. The **read_saif** command then strips off the prefix of a net name that matches *prefix* before deciding whether this net name is under the hierarchy specified by the **-ignore** option.

-exclude ***exclude_file_name***

Specifies the name of a file that contains a list of names to be ignored. The *exclude_file_name* option is used when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the **-exclude** option. As with the **-ignore** option, the ignore names are recognized after the **-strip_path** argument.

-derate_glitch ***value***

Specifies a default derating factor value to be used for inertial glitches in the SAIF file. If the **-derate_glitch** option is not specified, a default derating factor of 0.5 is used.

-quiet

Specifies quiet mode and for instance suppresses warnings on design objects in the SAIF file that could not be annotated on the design.

DESCRIPTION

The **read_saif** command reads a SAIF file and annotates the switching activity attributes *toggle_rate* and *static_probability* for nets, ports, and pins of the current design. The **update_power** command uses this information to calculate dynamic power values. If a particular object in the SAIF file cannot be found in the current design, the object is ignored and a warning message is provided. The **read_saif** command returns 1 if at least one of the objects in the file is successfully annotated. Otherwise, it returns 0.

To identify the instance (and corresponding subinstances) you want to annotate, use the **-strip_path** option. To annotate switching information about a particular instance, use the **-path** option. To specify a default derating factor other than 0.5, use the **-derate_glitch** option.

If a SAIF file comes from RTL simulation, the **read_saif** command tries to automatically locate the name mapping between the objects in the SAIF file and in the gate level netlist. The name mapping rules can be set by the **set_rtl_to_gate_name** command. During the average power calculation, the **update_power** command also automatically propagates the switching activity for those unannotated nets.

EXAMPLES

The following example annotates switching activity for the current design in the pt_shell, for a design instantiated as Test/U1 in the simulation testbench.

```
pt_shell> read_saif file -strip_path Test/U1
```

The following examples annotate the instance U10 under current design.

```
pt_shell> read_saif -file -strip_path Test/U1 -path U10
```

The following examples demonstrate annotation of multiple designs; the design "name1" is instantiated as Test1/U1, and design "name2" is instantiated as Test1/U2.

```
pt_shell> current_design name1

pt_shell> read_saif file_1 -strip_path Test1/U1

pt_shell> current_design name2

pt_shell> read_saif file_2 -strip_path Test1/U2
```

SEE ALSO

```
get_switching_activity(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_rtl_to_gate_name(2)
set_switching_activity(2)
update_power(2)
```

read_sdc

Reads in a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
int read_sdc
   [-echo]
   [-syntax_only]
   [-version sdc_version]
   file_name
```

Data Types

```
sdc_version          string
file_name            string
```

ARGUMENTS

-echo

Indicates that each constraint is to be echoed as it is executed.

-syntax_only

Indicates that SDC script is processed only to check the syntax and semantics of the script.

-version *sdc_version*

Specifies the version of SDC to which the file conforms. Allowed values are *1.0*, *1.1*, *1.2*, *1.3*, *1.4*, *1.5*, *1.6*, *1.7*, *1.8*, *1.9*, *2.0* and *latest* (the default).

file_name

Specifies the name of the file that contains the SDC script to be read.

DESCRIPTION

The **read_sdc** command reads in a script file in Synopsys Design Constraints (SDC) format that contains commands for use by PrimeTime or by Design Compiler in its Tcl mode. SDC is also licensed by external vendors through the Tap-in program. SDC formatted script files are Tcl scripts that use a subset of the commands supported by PrimeTime and Design Compiler.

By default, the **read_sdc** command executes the constraints as they are read. If there are errors part way

through the script, it is possible that some constraints are applied, and some are not. You can use the *-syntax_only* option to simply check the script without applying any constraints. This option also verifies conformance to the specified SDC version. If there are any errors during the checking phase, the result of the **read_sdc** command is 0.

Like the **source** command, the **read_sdc** command is sensitive to variables that control script execution (for example, the **sh_continue_on_error** and **sh_script_stop_severity** variables). The **read_sdc** command uses the **sh_source_uses_search_path** variable to determine if the **search_path** command is used to find the script.

The **read_sdc** command supports several file formats. The file can be a simple ASCII script file, an ASCII script file compressed with gzip, or a Tcl byte code script, created by TclPro Compiler.

Note that SDC does not allow command abbreviation. Also note that the **exit** and **quit** commands do not cause the application to exit, but they do cause the reading of the SDC file to stop.

The latest SDC version supports the commands listed below. Those added for versions 1.3 and later are noted. Some constraints that PrimeTime ignores are not listed.

General Purpose Commands

```
list
expr
set
```

Object Access Functions

```
all_clocks
all_inputs
all_outputs
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator
```

Basic Timing Assertions

```
create_clock
create_generated_clock          (1.3)
set_clock_gating_check
set_clock_latency
set_clock_transition
set_clock_uncertainty
set_data_check                  (1.4)
set_false_path
set_input_delay
set_max_delay
set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock
set_min_pulse_width             (2.0)
```

Secondary Assertions

```
set_disable_timing
set_max_time_borrow
set_timing_derate                (1.5)
```

Environment Assertions

```
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_load
set_logic_zero
set_logic_one
set_logic_dc
set_max_area
set_max_capacitance
set_max_fanout
set_max_transition
set_min_capacitance
set_min_fanout
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
```

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* in Documentation on the Web, which is available through SolvNet at the following address: http://solvnet.synopsys.com/DocsOnWeb

Using some of these commands is restricted when reading SDC. In some cases, some command options are not allowed. In some applications, some of the commands are not supported. For example, PrimeTime does not support the **set_logic*** commands. These commands are ignored when loaded in with the **read_sdc** command and a message appears (for an example, see the EXAMPLES section). If an unsupported command is located by the **read_sdc** command, it appears as an unknown command along with a message. The same condition exists for command options. If an option is not supported by SDC, a message is issued indicating that condition. However, if the option is unknown, a different message is issued.

Note that although the **create_generated_clock** command is supported (beginning with version 1.3), there is no provision for the **get_generated_clocks** shortcut, which is available in PrimeTime and Design Compiler. Generated clocks are simply clocks with some additional attributes and in SDC, they are retrieved through the **get_clocks** command.

The following features are added for SDC Version 1.5 or later:

- The **set_clock_latency** command with the *-clock* option
- The **set_timing_derate** command with all of the options
- The **set_max_transition** command with the *-clock_path -data_path -rise*, and *-fall* options
- The **set_clock_uncertainty** command with the *-rise*, *fall*, *from*, and *to* options
- The **set_operating_conditions** command with the *-object_list* option
- Synonyms for all **get_object** commands
- The **get_objects -nocase** command support independently with the *-regexp* option

- Support for the **get_objects -regexp** command
- The **get_objects -of_objects** support for the **get_cells get_nets**, and **get_pins** commands.

The following features are added for SDC Version 2.0 or later:

- The *-reference_pin* option for the commands **set_input_delay** and **set_output_delay**.
- The **set_min_pulse_width** command with the *-high* and *-low* options.

EXAMPLES

The following example reads the SDC script top.sdc.

```
pt_shell> read_sdc top.sdc -version 1.2
Reading SDC version 1.2...
1
```

The following example reads the SDC script top2.sdc, The script contains commands not supported by SDC, and CMD-005 messages are generated for those commands.

```
pt_shell> read_sdc -syntax_only top2.pt -version 1.2
Checking syntax/semantics using SDC version 1.2...
Error: Unknown command 'link_design' (CMD-005)
** Completed syntax/semantic check with 1 error(s) **
0
```

The following example shows the result when an unsupported command is in an SDC file. One SDC message is generated per instance of an unsupported command. At the end of the read, a summary of all unsupported commands in the file is generated. This SDC file was read into PrimeTime.

```
pt_shell> read_sdc t2.sdc -version 1.2
Reading SDC version 1.2...
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
Warning: Constraint 'set_logic_zero' is not supported by pt_shell. (SDC-3)
Warning: Constraint 'set_logic_one' is not supported by pt_shell. (SDC-3)

Summary of unsupported constraints:
Information: Ignored 1 unsupported 'set_logic_one' constraint. (SDC-4)
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (SDC-4)
1
```

The following example is a script that is not SDC-compliant, because it contains unsupported commands or command options.

```
current_design TOP
set_resistance -value 12.2 [get_nets i1t2]
```

The following example shows the output generated by the **read_sdc** command when reading the previous script. In SDC, the **current_design** command can be used only to get the current design; no arguments are allowed. Also, there is no *-value* option for the **set_resistance** command, so an appropriate message is generated.

```
pt_shell> read_sdc t3.tcl -echo -version 1.2
Reading SDC version 1.2...
current_design TOP
Error: extra positional option 'TOP' (CMD-012)
set_resistance -value 12.2 [get_nets i1t2]
Error: unknown option '-value' (CMD-010)
```

```
0
```

The following example shows the message generated when an option is supported by the command but not by SDC.

```
Information: The '-value' option for set_resistance is unsupported. (CMD-038)
```

SEE ALSO

```
source(2)
write_sdc(2)
search_path(3)
sh_continue_on_error(3)
sh_script_stop_severity(3)
sh_source_uses_search_path(3)
```

read_sdf

Reads leaf cell and net timing information from a file in Standard Delay Format (SDF) and uses that information to annotate the current design.

SYNTAX

```
string read_sdf
   [-path path_name]
   [-load_delay cell | net]
   [-analysis_type single | bc_wc | on_chip_variation]
   [-min_file min_fname]
   [-max_file max_fname]
   [-type sdf_min | sdf_typ | sdf_max]
   [-min_type sdf_min | sdf_typ | sdf_max]
   [-max_type sdf_min | sdf_typ | sdf_max]
   [-cond_use min | max | min_max]
   [-syntax_only]
   [-strip_path prefix_path]
   [-verbose]
   [-worst]
   file_name
```

Data Types

```
path_name       string
min_fname       string
max_fname       string
prefix_path     string
file_name       string
```

ARGUMENTS

-path *path_name*

Specifies the path from the current design to the subdesign for which the timing file has been created.

-load_delay cell | net

Specifies whether to read cell (the default) or net load delays in the timing file. The load delay is the portion of cell delay arising from the capacitive load of the net driven by the cell.

-analysis_type single | bc_wc | on_chip_variation

Use this option only if you have not already set an analysis type with the **set_operating_conditions -analysis_type** command. The available values are **single bc_wc**, and **on_chip_variation**. If you are in min_max mode, the default is the **bc_wc** value. The **single** value indicates that only one operating condition is to be used.

Specifying either the **bc_wc** or **on_chip_variation** values switches to min_max mode and causes both minimum and maximum delays to be read from the SDF file. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, the **-analysis_type bc_wc | on_chip_variation** option reads the *sdf_min* and *sdf_max* delays, respectively. To change this, use the **-min_type** and **-max_type** options.

-min_file *min_fname*

Specifies the file from which minimum delay timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0. Use this option only if the minimum and maximum delays are in two separate SDF files.

-max_file *max_fname*

Specifies the file from which maximum delay timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0. Use this option only if the minimum and maximum delays are in two separate SDF files.

-type sdf_min | sdf_typ | sdf_max

Specifies the SDF triplet delay value that is read from the SDF file: **sdf_min**, **sdf_typ**, or **sdf_max** (the default). Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max).

Note: If you use the **-type** option while in min/max mode (for example, if you use the **-operating_conditions bc_bw | on_chip_variation** option), a single value is annotated onto both minimum and maximum values of an arc.

-min_type sdf_min | sdf_typ | sdf_max

Specifies the SDF triplet delay value that is read from the SDF file for minimum delay: **sdf_min** (the default), **sdf_typ**, or **sdf_max**. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). Use this option only with the **-analysis_type bc_wc | on_chip_variation** option.

-max_type sdf_min | sdf_typ | sdf_max

Specifies the SDF triplet delay value that is read from the SDF file for maximum delay: **sdf_min**, **sdf_typ**, or **sdf_max** (the default). Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). Use this option only with the **-analysis_type bc_wc | on_chip_variation** option.

-cond_use min | max | min_max

Use this option only if the SDF file includes some conditional delays using the SDF construct COND, and if the Synopsys library in use does not specify conditional delays. The available values are **min**, **max**, **min_max**. The **min** value indicates that the minimum of all conditional delays is used to annotate the corresponding timing arc. The **max** value indicates to use the maximum; The **min_max** value indicates min_max operating conditions. The minimum of all conditional delays is used for the minimum operating condition, and the maximum of all conditional delays is used for the maximum operating condition. You cannot use the **min_max** value with a single operating condition; you must be in the min_max mode.

-syntax_only

Verifies the SDF syntax. No timing annotation is performed; only syntax is processed. Use this option to verify that your SDF syntax is correct.

-strip_path *prefix_path*

Specifies a prefix path that is stripped from all SDF objects. Such a prefix path is usually a result of generating an SDF file for a subdesign and using this subdesign as the current design.

-verbose

Use this option to enable execution of the **report_annotated_delay** and **report_annotated_check**

commands after reading SDF.

-worst

Indicates that the **read_sdf** command annotates the current design only with delays worse than the current annotated delays; applies to annotated net and cell delays and annotated timing checks. The worst delay is defined as the most pessimistic delay. This means PrimeTime annotates the minimum of minima and maximum of maxima values.

file_name

Specifies the file from which timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0.

DESCRIPTION

The **read_sdf** command reads from a disk file leaf cell and net timing information and uses it to annotate the current design. The timing file must be in SDF format v1.0, v2.0, v2.1 or v3.0. The file can be a simple ASCII file, or it can be compressed with gzip. Instance-specific pin-to-pin cell and net delays are read from the SDF file and annotated on the current design. When you specify the **-path** option, the **read_sdf** command annotates the current design with information from a timing file created for an instance of a subdesign of the current design. When you specify a subdesign, you cannot use the net delays to the ports of the subdesign to annotate the current design.

The load delay, also known as extra source gate delay, is that portion of the cell delay caused by the capacitive load of the driven net. Some delay calculators consider the load delay part of the net delay. Others consider the load delay part of the cell delay. By default, the **read_sdf** command assumes the load delay is included in the cell delay in the timing file being read. The **-load_delay** option allows you to inform the **read_sdf** command if your timing file includes the load delay in the net delay rather than in the cell delay.

Setup, hold, recovery, removal, period, width, nochange, and skew timing checks, when present in the timing file, are used to annotate the current design. Prior to SDF version 3.0, the SDF construct for removal is HOLD.

Instance names in the design must match instance names in the timing file. For example, if the timing file is created from a design using VHDL naming conventions, the design must use VHDL naming conventions.

To remove delays read and annotated with the **read_sdf** command, use the **reset_design** or **remove_annotated_delay** command. To remove timing checks annotated with the **read_sdf** command, use the **remove_annotated_check** command.

Only partial support for SDF version 3.0 is available. Supported v 3.0 constructs are: REMOVAL, RECREM, RETAIN, and CONDELSE.

Note that the **read_sdf** command does not support hierarchical SDF annotation. All pins must be specified at the leaf cell level. It is not possible to specify hierarchical pins.

When multiple annotations against the same arc are present in the annotation file, the last annotation will take precedence. If the **-worst** option is specified, each annotation is applied only if it is worse than the existing timing value for that arc.

Also note that if the **timing_use_zero_slew_for_annotated_arcs** variable is set to its default **auto** value and at least 95% of delay arcs are annotated, the pure SDF flow is switched on automatically. This results in faster performance of the **update_timing** command, at the expense of no longer calculating the slews on the load pins of annotated arcs. For more information, see the **timing_use_zero_slew_for_annotated_arcs** man page.

EXAMPLES

The following example reads the adder.sdf timing file from the disk and uses it to annotate the timing on the current design. The timing file contains load delays included in the cell delays.

```
pt_shell> read_sdf -load_delay cell  adder.sdf
```

The following example reads timing information for instance u1 of design MULT16 from the mult16_u1.sdf disk file that contains the timing for the instance u1 of design MULT16. The timing is annotated on the design MY_DESIGN. In this case, load delay is included in the net delays.

```
pt_shell> current_design MY_DESIGN
pt_shell> read_sdf -load_delay net -path u1 mult16_u1.sdf
```

The following example reads timing information and annotates the current design with the maximum timing when the timing file has different timing conditions for the same pin pair. The load delay is assumed to be included in the cell delay in this example.

```
pt_shell> remove_annotated_delay
pt_shell> read_sdf -cond_use max fname1.sdf
```

The following example reads minimum and maximum timing information and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively. When reporting minimum delay, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delay, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc fname2.sdf
```

The following example reads minimum and maximum timing information from two separate SDF files and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively. When reporting minimum delay, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delay, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc -min_file foo_min.sdf -max_file fname3_max.sdf
```

SEE ALSO

```
remove_annotated_check(2)
remove_annotated_delay(2)
report_annotated_check(2)
report_annotated_delay(2)
reset_design(2)
set_annotated_check(2)
set_annotated_delay(2)
write_sdf(2)
timing_use_zero_slew_for_annotated_arcs(3)
```

read_supply_parasitics

Reads RC parasitics for supply nets from the extraction data directory.

SYNTAX

```
status read_supply_parasitics
   supply_nets
   [-noexpand ]
```

Data Types

```
supply net collection supply_nets
```

ARGUMENTS

supply_nets

Lists the supply nets for which to read parasitics data. Each element of the list can be a supply net collection or a name pattern. The command automatically expands the supply net list to include the supply nets that are connected to those listed nets through multithreshold-CMOS switch cells (also known as power management or PM cells). To suppress the expansion, use the -noexpand option.

-noexpand

Prevents the *supply_net_list* from being expanded to include the related supply nets that are indirectly connected to the listed nets through multithreshold-CMOS switch cells. By default, the list of supply nets is expanded.

DESCRIPTION

The **read_supply_parasitics** command reads RC parasitics for the supply nets from the extraction data directory generated during the **extract_supply_parasitics** command run. The default directory name is RAIL_DATABASE/pgext/para.

The **read_supply_parasitics** reads either RC-based or R-only parasitics data, depending on the type of extraction is performed with the **extract_supply_parasitics** command. If the flow is a reuse flow without extraction, the command reuses RC-based parasitics by default. However, if no RC-based parasitics data is found in the RAIL_DATABASE/pgext/para directory, the command reads R-only parasitics data instead.

The command waits for the completion of the extraction if the -nowait option is used with the **extract_supply_parasitics** command.

The list of supply nets for which to read the RC parasitics is automatically expanded to include the supply nets that

are connected to the specified nets through multithreshold-CMOS switch cells. This automatic supply net expansion behavior saves you from identifying the full set of related supply nets yourself. You can suppress this auto expansion by using the -noexpand option.

EXAMPLES

The following example reads the parasitic RC values for the *VDD* and *VSS* nets contained in the ./RAIL_DATABASE/pgext/para directory.

```
pr_shell> read_supply_parasitics {VDD VSS}
```

SEE ALSO

current_rail_result(2)
extract_supply_parasitics(2)
report_supply_parasitics(2)
set_extract_supply_net_options(2)

read_vcd

Specifies the switching activity information generated by simulation for use in power calculation. Internally, non-VCD format switching activity is converted to VCD.

SYNTAX

```
status read_vcd
   [-pipe_exec pipe]
   [-path path]
   [-strip_path strip_path]
   [-zero_delay]
   [-rtl]
   [-format verilog | vhdl | systemverilog]
   [-time window_list]
   [-when condition]
   [-converter_options option_list]
   file_name
```

Data Types

```
pipe            string
path            string
strip_path      string
window_list     list
condition       string
option_list     list
file_name       string
```

ARGUMENTS

-pipe_exec *pipe*

Specifies a shell command that is used to generate the VCD file *file_name*. This option invokes the command and directly pipe the output VCD file to PrimeTime PX. In other words, the simulation and power analysis are in parallel run. No VCD disk file is generated at all.

-path *path*

Specifies a relative path from the current design to the hierarchical low-level design for which the VCD file has been created. By default, absolute path names are used. Use this option if the VCD file refers to an object in a hierarchy. Do not use this option if the VCD file refers to an absolute path.

-strip_path *strip_path*

Specifies a path prefix that is to be stripped from all the object names read from the VCD file. This option is applied to strip the testbench and instance path from the VCD file.

-zero_delay

Specifies that he VCD file comes from a zero delay simulation.

-rtl

Specifies the VCD file comes from an RTL simulation. Typically, this means that most nets in the design are not directly annotated from the VCD, because combinational nets are not included in the RTL simulation. The **set_rtl_to_gate_name** name mapping command is honored only if this option is set. Also, in the **time_based** power analysis mode, event propagation occurs during power analysis only if the **-rtl** option is set when the **read_vcd** command was issued.

-format verilog | vhdl | systemverilog

Specifies the language format for names in the VCD. Available options are **verilog** (the default), **vhdl**, and **systemverilog**.

-time *window_list*

Specifies time windows in which the activities are counted for power calculation. The option accepts an even number list of float values in increasing order. For example, **-time {10 20 50 70}** specifies the time window [10ns, 20ns] and [50ns, 70ns]. If you do not know the end of simulation time, use a negative number in the **-time** option to indicate the end of the simulation. For example, **-time {10 -1}** specifies the activity time window from 10 ns to the end of simulation time in the VCD file. The default activity time window is the whole simulation time in VCD file. This option applies to averaged, time-based power analysis and statistical leakage power variation analysis. The time is automatically adjusted by the tool according to the timescale in the VCD file or the **-waveform_interval** value in the **set_power_analysis_option** command. For example, if the **-waveform_interval** is 10, **-time {13 33}** becomes **-time {10 30}**.

-when *condition*

Specifies a Boolean condition to determine which simulation times from the VCD should be considered for power analysis. The Boolean condition is a string using names of individual signals in VCD (either annotated to design objects or not annotated), and also operators. Similar to the **-time** option, the **-when** option selects portions of a VCD for analysis. However, with the **-when** option, the times selected are chosen automatically based on logical values in the VCD. During simulation times when the Boolean condition is **true**, the events and power are included in power analysis up to and including the timestamp when the condition becomes false. For simulation times when the Boolean condition is false, the events and power are excluded from analysis up to and including the timestamp when the condition becomes true. The feature can be used in both **averaged** and **time_based** power analysis modes.

-converter_options *option_list*

Specifies options for external VCD converters. If the **-converter_options** is specified with list of options, the specified options will be used in conjunction with the external VCD converters, vpd2vcd or fsdb2vcd.

file_name

Specifies the switching activity file name to be read. If the *file_name* ends with the .gz, .Z, .bz or .bz2, .vpd, or .fsdb extension, it is read as a gzipped, compressed, bzip2ed, VPD (VCD Plus), or FSDB file, respectively; otherwise, it is assumed to be in the VCD format.

DESCRIPTION

The **read_vcd** command specifies the switching activity file generated by simulation to be used for time-based power calculation.

To read in a gzipped VCD file, a compressed VCD file, a BZIP2 file, a VPD file, and an FSDB file, your PATH should

include gunzip, uncompress, bunzip, vcs, and fsdb2vcd (a utility from FSDB inventor Novas http://www.novas.com). The VPD file is generated by VCS. So you should define the environment variable VCS_HOME to tell the tool where to find the VCS utility. If you need to invoke 64-bit VCS utility, you need to set the environment variable VCS_MODE_FLAG to 64. You can also define the environment variable NOVAS_HOME to tell the tool where to find the fsdb2vcd executable.

The **-pipe_exec** option allows PrimeTime PX to read in a VCD file directly from simulator, thus avoiding the need to generate a large VCD file. The VCD file name in this scenario serves as a pipe name. After simulation, the VCD file does not exist. There is no need to modify the existing testbench. The file name is treated like a normal VCD output.(inserting $dumpfile, $dumpvars, etc.) Nor is there any need to invoke the simulator. PrimeTime PX automatically runs the simulation and directly read in the VCD output from the simulator.

When a VCD file comes from a zero delay gate level simulation; most of events happen at clock edges. This leads to a very large unrealistic peak power. With the **-zero_delay** option, power is averaged over each clock cycle. This requires you to specify the Synopsys Design Constraint (SDC) file, at least clocks and transition time at primary inputs have to be defined.

If a VCD file comes from RTL simulation, the **-rtl** option should be used. In this case, the **read_vcd** attempts to find the name mapping between the objects in the VCD file and in the gate level netlist automatically. The name mapping rules can be set by the **set_rtl_to_gate_name** command. During the average power calculation, the **update_power** command also automatically propagates the switching activity for those unannotated nets.

Behavior For Different Power Modes

Because of the way analysis is performed in different power analysis modes, the **read_vcd** command has different behavior depending on the power analysis mode. To specify the power analysis mode, set the **power_analysis_mode** variable to **averaged**, **time_based**, or **leakage_variation**.

In the **averaged** and **leakage_variation** modes, power analysis is performed using toggle rates. In these modes the input activity file is converted to toggle rates and static probabilities, which are annotated onto the design.

In the **time_based** mode, power analysis is performed for individual events in the activity file. The **read_vcd** command is a required part of this flow. In this flow, the header of the activity file is read when the **read_vcd** command is issued, to determine what portion of the design is annotated. The remainder of the activity file is not read until power analysis is performed.

In the **time_based** mode, if you use the **-rtl** option, the tool uses name mapping (see the **set_rtl_to_gate_name** command), and power analysis uses zero delay simulation to create events on unannotated nets. Without the **-rtl** option, name mapping is not used and no events is simulated. It is intended that RTLVCD be in the input if the option is used. In this case, sequential cells, primary inputs, and black box outputs are typically annotated, while combinational nets are not annotated. Events on these unannotated nets must be produced via simulation.

EXAMPLES

The following example reads the VCD file dump.vcd from the disk and uses the first 100 ns of the events in the file.

```
pt_shell> read_vcd dump.vcd -time {0 100}
```

The following example runs VCS and pipes the VCD file dump.vcd directly to PrimeTime PX:

```
pt_shell> read_vcd -pipe_exec "vcs -R tb.v design.v" dump.vcd
```

The following example reads a VCD file and excludes time periods when the specified condition is false.

```
pt_shell> read_vcd dump.vcd -when {net125 && net126}
```

SEE ALSO

read_db(2)
read_verilog(2)
read_vhdl(2)
set_power_analysis_options(2)
set_rtl_to_gate_name(2)
power_analysis_mode(3)

read_verilog

Reads in one or more design or library files in Verilog format.

SYNTAX

```
status read_verilog
   [-dirty_netlist]
   [-allow_black_box]
   [-verbose]
   [-bus_direction_for_undefined_cell connection | msb | lsb]
   [-keep_module keep_module_list]
   [-top top_module_name]
   [-cell cell_name]
   [-netlist | -rtl]
   verilog_files
```

Data Types

keep_module_list	list
top_module_name	string
cell_name	string
verilog_files	list

ARGUMENTS

-dirty_netlist

Controls whether the Verilog reader handles a dirty netlist. A dirty netlist is a Verilog netlist that contains objects that are inconsistent with the design libraries, such as floating pins, nets, and pins or ports. When the Verilog reader reads in a dirty netlist, it checks for the following missing or incomplete information issues warning messages, and then creates objects that allow the tool to continue:

- Missing reference cell in the reference library
  There is no reference cell available in the reference libraries and the definition is not in the input Verilog file.
- Port definition is inconsistent with the reference library
  There are additional ports present that are not available in the reference library.
- Port definition is inconsistent with the reference cell
  There are additional ports present that are not available in the reference cell definition.

-allow_black_box

Allows the tool to continue if a module cannot be found in the reference libraries or in the input Verilog file.

If you do not specify this option, **read_verilog** issues an error message and quits when it finds an undefined module.

-verbose

Displays verbose information including possible top-module names, black-box names, and module names with mismatched ports.

-bus_direction_for_undefined_cell connection | msb | lsb

Specifies the bus direction for an undefined cell. If an undefined module is instantiated, **read_verilog** determines the bus direction according to one of following three factors: connection, msb, or lsb. By default, it determines the bus direction from the connection.

-keep_module *keep_module_list*

Specifies the list of keep modules. For each module in the list, the Verilog reader does not expand it into the top cell. Instead, it expands it into a soft macro cell and instantiates it in the top cell. This is for the design planning top-down flow.

-top *top_module_name*

Specifies the top-module name. The top module is the module that is not instantiated by any other modules. Generally there is only one top module in a design and **read_verilog** can scan and identify the top module automatically. However, if the design has multiple top modules, you must use this option to specify one of the modules.

-cell *cell_name*

Specifies the cell name. The tool creates the design in the Milkyway database using the specified cell name. If not specified, the tool uses the name of the top module as the cell name.

verilog_files

Specifies the name of one or more Verilog files to be read.

-netlist

Specifies that the design being read is a structural or gate-level design. This option invokes a netlist reader optimized for gate-level designs. If any of the specified files contain RTL-level constructs, an error is issued. Use of the **-netlist** option might enable the tool to read larger gate-level designs than would otherwise be the case.

This option is mutually exclusive with the **-rtl** option.

-rtl

Specifies that the design being read is an RTL design. This option invokes the Presto RTL Verilog reader directly without attempting to determine if the specified files are gate-level netlists. For details, see the **hdlin_auto_netlist_reader** variable man page.

This option is mutually exclusive with the **-netlist** option.

DESCRIPTION

.prod icc

The **read_verilog** command loads in netlist information from one or more Verilog files. .prod syn The **read_verilog** command reads design information from Verilog files into memory.

If the automatic mode is on, the netlist reader (read with the **-netlist** option) is automatically invoked. If the input contains RTL constructs, the system automatically invokes the RTL reader (read with **-rtl** option).

This command is derived from **read_file -format verilog**. For more information, see the man page for the **read_file** command. .prod all

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

`read_file(2)`

redirect

Redirects the output of a command to a file.

SYNTAX

```
string redirect
   [-append]
   [-tee]
   [-file | -variable | -channel]
   [-compress]
   [-bg]
   target
   command_string
```

Data Types

target	string
command_string	string

ARGUMENTS

-append

Appends the output to the *target* argument.

-tee

Like the UNIX command of the same name, sends output to the current output channel as well as to the *target* argument.

-file

Indicates that the *target* argument is a file name, and redirection is to that file. This is the default. It is exclusive of the *-variable* and *-channel* options.

-variable

Indicates that the *target* argument is a variable name, and redirection is to that Tcl variable. It is exclusive of the *-file* and *-channel* options.

-channel

Indicates that the *target* argument is a Tcl channel, and redirection is to that channel. It is exclusive of the *-variable* and *-file* options.

-compress

Compresses when writing to file. If redirecting to a file, this option specifies that the output should be

compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with the *-append* option.

-bg

Indicates that the redirected command executes in the background and in parallel with other foreground commands downstream. PrimeTime returns immediately to execute the next command after this redirect command, unless the next command is **exit** or **quit**, which blocks the run until all background commands with the *-bg* option are finished. Note the *-bg* option must be used together with the *-file* option and no other options are allowed. Redirect command returns the PID of the background process when this option is specified. If the process cannot be successfully started, an empty string is returned.

target

Indicates the target of the output redirection. This is either a filename, Tcl variable, or Tcl channel depending on the command arguments.

command_string

The command to execute. Intermediate output from this command, as well as the result of the command, is redirected to the *target* argument. The *command_string* option should be rigidly quoted with curly braces.

DESCRIPTION

The **redirect** command performs the same function as the traditional UNIX-style redirection operators > and >>. The *command_string* option must be rigidly quoted (that is, enclosed in curly braces) in order for the operation to succeed.

Output is redirected to a file by default. Output can be redirected to a Tcl variable by using the *-variable* option, or to a Tcl channel by using the *-channel* option.

Output can be sent to the current output device, as well as the redirect target by using the *-tee* option. See the Examples section for an example.

The result of a **redirect** command which does not generate a Tcl error, is the empty string. Screen output occurs only if errors occurred during execution of the *command_string* option (other than opening the redirect file). When errors occur, a summary message is printed. See the Examples section for an example.

Although the result of a successful **redirect** command is the empty string, it is still possible to get and use the result of the command that you redirected. Construct a **set** command in which you set a variable to the result of your command. Then, redirect the **set** command. The variable holds the result of your command. See the Examples section for an example.

The **redirect** command is much more flexible than traditional Unix redirection operators. With the **redirect** command, you can redirect multiple commands or an entire script. See the Examples section for an example of how to construct such a command.

Note that the built-in **puts** Tcl command does not respond to output redirection of any kind. Use the built-in **echo** command instead.

The *-bg* option is introduced to enable parallel processing on multicore hardware for performance reasons. PrimeTime returns immediately to execute the next command after this redirect command, unless the next command is **exit** or **quit**, which blocks the process until all background commands launched with the *-bg* option previously are finished. Note the *-bg* option must be used together with the *-file* option and no other options are allowed.

It is also very important to understand that the *-bg* option for the **redirect** command is targeted for post-update reporting and analysis commands. Any command executed in the background causes changes to timing data and triggers incremental update, which result in severe error and stop of execution. In addition, it is advisable to intentionally block the foreground process if the command to be executed invalidates the existing timing analysis, such as the **update_timing**, **update_noise**, **remove_design** commands. Refer to the **parallel_execute** man page for more details. Use of the **parallel_execute** command is recommended whenever applicable to perform parallel execution of commands.

EXAMPLES

In the following example, the output of the plus procedure is redirected. The echoed string and the result of the plus operation is in the output file. Notice that the result was not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command created an error condition. The error message indicates that you can use the **error_info** command to trace the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
        Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

In this example, we explore the usage of results from redirected commands. Since the result of the **redirect** command for a command which does not generate a Tcl error is the empty string, use the **set** command to trap the result of the command. For example, assume that there is a command to read a file which has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
redirect p.out { read_a_file "a.txt" }
# Now what?  How can I redirect and use the result?
```

But if you set a variable to the result, then it is possible to use that result in a conditional expression and so forth.

```
redirect p.out { set rres [read_a_file "a.txt"] }
if { $rres == 1 } {
  echo "Read ok!"
}
```

The **redirect** command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single **redirect** command. This example with the **echo** command demonstrates this feature:

```
prompt> redirect p.out {
?         echo -n "Hello "
?         echo "world"
?       }
prompt> exec cat p.out
Hello world
prompt>
```

The **redirect** command allows you to tee output to the previous output device and also to redirect output to a variable. This example with the **echo** command demonstrates these features:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
          echo XXX
          redirect -variable y -append {
            echo YYY
            redirect -tee -variable z {
              echo ZZZ
            }
          }
        }
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ

prompt> echo $z
ZZZ
```

SEE ALSO

echo(2)
error_info(2)
set(2)
parallel_execute(2)

remove_annotated_check

Removes annotated timing checks from the design, either on specific cells, between specific pins, or on all cells in the current design.

SYNTAX

```
status remove_annotated_check
   [-all]
   [-from from_pins]
   [-to to_pins]
   [-rise]
   [-fall]
   [-clock clock_check]
   [-setup | -recovery]
   [-hold | -removal]
   [-nochange_low]
   [-nochange_high]
   [-width]
   [object_spec]
```

Data Types

```
from_pins       list
to_pins         list
clock_check     string
object_spec     list
```

ARGUMENTS

-all

Removes all annotated timing checks in the design. This option is exclusive of the **-from**, **-to**, and *object_spec* options. Options that specify the type of the timing check, such as **-setup** and **-hold** are ignored when you use the **-all** option.

-from *from_pins*

Specifies a list of leaf cell pins that are the startpoints of the timing arcs for which annotated checks are to be removed. You must use the **-from** option with the **-to** option, and you cannot combine these options with the *object_spec* option.

-to *to_pins*

Specifies a list of leaf cell pins that are the endpoints of the timing arcs for which annotated checks are to be removed. You must use the **-from** option with the **-to** option, and you cannot combine these options with the *object_spec* option.

-rise

Specifies that the timing check is for the rise transition on the constrained (the data **-to** pin). The **-rise** and **-fall** options are mutually exclusive. If you do not specify either the **-rise** or **-fall** option, both values are removed.

-fall

Specifies that the timing check is for the fall transition on the constrained (the data **-to** pin). The **-rise** and **-fall** options are mutually exclusive. If you do not specify either the **-rise** or **-fall** option, both values are removed.

-clock *clock_check*

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are removed. The valid values for the *clock_check* options are *rise* or *fall*.

-setup

Removes only setup timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-setup** option with the **-recover** option.

-recovery

Removes only recovery timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-setup** option with the **-recover** option.

-hold

Removes only hold timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-hold** option with the **-removal** option.

-removal

Removes only removal timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-hold** option with the **-removal** option.

-nochange_low

Removes only nochange timing check information against data low information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-nochange_low** option with the **-nochange_high** option.

-nochange_high

Removes only nochange timing check information against data high information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the **-nochange_low** option with the **-nochange_high** option.

-width

Removes only minimum pulse width check annotation.

object_spec

Removes annotated timing checks from the specified list of leaf cells. You cannot combine this option with the **-from** and **-to** options.

DESCRIPTION

This command removes annotated timing checks from the design, either on specific cells, between specific pins, or on all cells in the current design. Timing checks are annotated either from a Standard Delay Format (SDF) file by using the **read_sdf** or **set_annotated_check** command. Timing check types include setup, hold, recovery, removal, and nochange.

You can use the **remove_annotated_check** command in the following ways:

- You can remove all annotated checks from the entire design by using the **-all** option; in this case, you cannot remove a specific check. The command ignores options that specify the type of the timing check such as, the **-setup** and **-hold** options when you use the **-all** option.
- You can remove annotated checks from a list of cells by using the *object_spec* option. Without specifying any timing check type options, the command removes all annotated checks from each cell in the list. By using options that specify the type of the timing check, such as the **-setup** and **-hold** option you can remove only specific checks. You can use this method to remove all checks of a certain type from the entire design. For an example, see the EXAMPLES section.
- You can remove annotated checks between specific pins by using the **-from** and **-to** options. Each from/to pair must be on the same cell. Without specifying any timing check type options, the command removes all annotated checks from each from/to pair. By using options that specify the type of the timing check,, such as the **-setup** and **-hold** options, you can remove only specific checks.

EXAMPLES

The following example removes all annotated cell timing checks from the current design.

```
pt_shell> remove_annotated_check -all
1
```

The following example removes only setup checks from all cells in the current design.

```
pt_shell> remove_annotated_check -setup [get_cells *]
1
```

The following example removes checks between pins. It also shows the error condition when the pins are not on the same cell and the warning when there are no annotated checks.

```
pt_shell> remove_annotated_check -from ffb/CP -to ffa/D

Error: Cannot remove annotated check from 'ffb/CP' to 'ffa/D':
       pins are on different cells (PTE-032)
0
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D -setup -hold
1
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D
Warning: No annotated timing checks were removed. (PTE-031)
0
```

SEE ALSO

```
read_sdf(2)
report_annotated_check(2)
report_annotated_delay(2)
reset_design(2)
set_annotated_check(2)
set_annotated_delay(2)
```

remove_annotated_clock_network_power

Removes the annotated power on clock networks.

SYNTAX

```
status remove_annotated_clock_network_power
   [-clock domain_clock]
```

Data Types

```
domain_clock     string
```

ARGUMENTS

-clock *domain_clock*

Specifies the clock of the related clock network on which the annotated power values are removed.

DESCRIPTION

The **remove_annotated_clock_network_power** command removes the annotated power information specified by the **set_annotated_clock_network_power** command on the current design. If the *-clock* option is used, only the annotated clock network power for the specific clock domain is removed; otherwise, the annotated power values for all clock domains are removed. If the power is not annotated for the specified clock domain, a PWR-294 error message is issued and the command fails.

EXAMPLES

In the following example, the **remove_annotated_clock_network_power** command is used before the **report_power** command and after the **set_annotated_clock_network_power** command. The output from the **report_power** command is not affected.

```
pt_shell> set_annotated_clock_network_power -internal 1.0e-03 -switching 2.0e-03
pt_shell> remove_annotated_clock_network_power
pt_shell> report_power
```

SEE ALSO

```
report_power(2)
set_annotated_clock_network_power(2)
```

remove_annotated_delay

Removes annotated delays from the design, either on specific cells or nets, between specific pins, or all annotated delays in the design.

SYNTAX

```
status remove_annotated_delay
   [-all]
   [-from from_list]
   [-to to_list]
   [object_spec]
```

Data Types

```
from_list        list
to_list          list
object_spec      list
```

ARGUMENTS

-all

Removes all annotated delays in the design. This option is exclusive of the **-from**, **-to**, and *object_spec* options.

-from *from_list*

Specifies a list of pins or ports that are the startpoints of the timing arcs for which annotated delays are removed. You cannot combine this option with the *object_spec* option.

-to *to_list*

Specifies a list of pins or ports that are the endpoints of the timing arcs for which annotated delays are removed. You cannot combine this option with the *object_spec* option.

object_spec

Removes all annotated delays from the specified list of leaf cells or nets. You cannot combine this option with the **-from** and **-to** options.

DESCRIPTION

The **remove_annotated_delay** command removes annotated cell and net delays from the current design. Delays are annotated either from an SDF file (using the **read_sdf** command) or by using the

set_annotated_delay command.

There are several ways to use the **remove_annotated_delay** command.

- You can remove all annotated delays from the entire design using the **-all** option.
- You can remove all annotated delays from a list of cells or nets using the *object_spec* option.
- You can remove annotated delays between specific pins using the **-from** and **-to** options.

EXAMPLES

The following example removes all annotated net and cell delays from the current design.

```
pt_shell> remove_annotated_delay -all
1
```

The following example removes annotated delays from specific cells.

```
pt_shell> remove_annotated_delay [get_cells u1*]
1
```

The following example removes annotated delays between pins. Also shown is the warning when there are no annotated delays between the specified pins.

```
pt_shell> remove_annotated_delay -from ffb/Q
1
pt_shell> remove_annotated_delay -from ffb/Q -to u1/A
Warning: No annotated delays from 'ffb/Q' to 'u1/A'. (PTE-030)
0
```

SEE ALSO

```
read_sdf(2)
remove_annotated_check(2)
report_annotated_check(2)
report_annotated_delay(2)
reset_design(2)
set_annotated_check(2)
set_annotated_delay(2)
```

remove_annotated_mtcmos_switch_cell

Removes user annotations from multithreshold-CMOS switch cells and reverts the *on_state_resistance* and *off_state_leakage* parameters to the default value. The user annotation was previously applied by the **set_annotated_mtcmos_switch_cell** command.

SYNTAX

```
status remove_annotated_mtcmos_switch_cell
   object_list
   -on_state_resistance | -off_state_leakage
   [-pin_state pin_state | -name name]
```

Data Types

```
list object_list
list pin_state
string name
```

ARGUMENTS

object_list

A list of multithreshold-CMOS switch cell objects for which to remove the previously applied user annotation. Each element in the list can be a collection of multithreshold-CMOS switch cell objects or a cell object name pattern.

-on_state_resistance

Indicates that on-state resistance is to be removed.

-off_state_leakage

Indicates that off-state leakage is to be removed.

-pin_state *pin_state*

Specifies a list of pin states for which the annotation is to be removed from the multithreshold-CMOS switch cell. When specified, the command removes annotations for all on-states of the multithreshold-CMOS switch cell in the library. If a pin name specified in the list does not match any pin in the cell, the command run will fail. If the multithreshold-CMOS switch cell consists of more pins than those specified in the pin state list, state X (neither 0 nor 1) will be assigned to pins whose states are not set in the list. If the pin state does not match any on-state in the library for the multithreshold-CMOS switch cell, the command run will fail.

-name *name*

Specifies a pin state name for which the annotated value is to be removed. If the specified name does not exist, the command run will fail.

DESCRIPTION

The command removes the user annotation that is applied to the specified multithreshold-CMOS switch cell objects by the **set_annotated_mtcmos_switch_cell** command.

When the applied *on_state_resistance* is to be removed, the command reverts it to the default on-state resistance for the multithreshold-CMOS switch cell object. Typically the default resistance is derived from the characterization data of the library cell that contains the multithreshold-CMOS switch cell. The lib_cell *on_state_resistance* attribute provides the default resistance value, and the cell *on_state_resistance* attribute provides the current resistance. If the characterization data of the library is not available, the *on_state_resistance* value is set to infinite, which means the switch cell is turned off.

When the applied *off_state_leakage* is removed, the command reverts it to the default off-state leakage for the multithreshold-CMOS switch cell object. Typically the default leakage is derived from the characterization data of the library cell that contains the multithreshold-CMOS switch cell. The lib_cell *off_state_leakage* attribute provides the default leakage value, and the cell *off_state_leakage* attribute provides the current leakage. If the characterization data of the library is not available, the *off_state_leakage* value is set to 0.

You do not need to remove user-annotated on- or off-state for the multithreshold-CMOS switch cell object, because you can set the state to be off to disconnect two PG networks through the multithreshold-CMOS switch cell.

You cannot remove a user annotation that is not set by the **set_annotated_mtcmos_switch_cell** command. If this is attempted, the command is ignored and a success code is returned.

EXAMPLES

The following example gets, sets and removes the resistance values that are applied by the **set_annotated_mtcmos_switch_cell** command.

```
pr_shell> get_attribute [get_cell -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ on low "i 1" vdd vvdd 0.52 10.0 }
pr_shell> get_attribute [get_lib_cell */* -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ low "i 1" vdd vvdd 10.0 }
pr_shell> set_annotated_mtcmos_switch_cell 2.0 \
                  [get_cell -filter is_mtcmos_switch_cell] \
                  -name low
1
pr_shell> get_attribute [get_cell -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ on low "i 1" vdd vvdd 2.0 10.0 }
pr_shell> get_attribute [get_lib_cell */* -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ low "i 1" vdd vvdd 10.0 }
pr_shell> remove_annotated_mtcmos_switch_cell -on_state_resistance \
                  [get_cell -filter is_mtcmos_switch_cell]
1
pr_shell> get_attribute [get_cell -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ on "i" "" vdd vvdd "" 10.0 }
pr_shell> get_attribute [get_lib_cell -filter is_mtcmos_switch_cell] \
```

```
                  on_state_resistance
{ low "i 1" vdd vvdd 10.0 }
pr_shell> set_annotated_mtcmos_switch_cell off -state \
                  [get_cell -filter is_mtcmos_switch_cell]
1
pr_shell> get_attribute [get_cell -filter is_mtcmos_switch_cell] \
                  on_state_resistance
{ off low "i 1" vdd vvdd "" "" }
```

SEE ALSO

```
set_annotated_mtcmos_switch_cell(2)
get_cell(2)
report_mtcmos_switch_cell(2)
get_lib_cell(2)
```

remove_annotated_parasitics

Removes all annotated parasitics from nets of the current design.

SYNTAX

```
status remove_annotated_parasitics
   [-all]
   [net_list ]
```

Data Types

```
net_list          list
```

ARGUMENTS

-all

Indicates that all annotated nets in the design are to be removed, which is the default. The **-all** and *net_list* options are mutually exclusive.

net_list

Specifies a list of nets from which annotated parasitics are to be removed. By default, all annotated parasitics are removed from the design. The **-all** and *net_list* options are mutually exclusive.

DESCRIPTION

This command removes all net parasitics annotated on nets of the current design. The annotated parasitics are usually added using the **read_parasitics** command to read a SPEF or standard parasitic format (SPF) file and annotate the parasitics on nets of the current design. The **remove_annotated_parasitics** command also removes the annotated parasitics completed by the **complete_net_parasitics** command.

There are two ways to use the **remove_annotated_parasitics** command.

- You can remove all annotated parasitics from the entire design by using the **-all** option or by executing the command without options.
- You can remove all annotated parasitics from a list of nets using the *net_list* option.

EXAMPLES

The following example removes all annotated net parasitics from the current design.

```
pt_shell> remove_annotated_parasitics -all
1
```

The following example removes annotated parasitics from the net "CLK".

```
pt_shell> remove_annotated_parasitics [get_nets CLK]
1
```

SEE ALSO

```
read_parasitics(2)
report_annotated_parasitics(2)
reset_design(2)
```

remove_annotated_power

Removes previously annotated power from unresolved black box cells or leaf cells.

SYNTAX

```
status remove_annotated_power
   -all
   cell_list
```

Data Types

```
cell_list        list
```

ARGUMENTS

-all

Specifies that all annotated powers in the design are to be removed. The **-all** and *cell_list* options are mutually exclusive.

cell_list

Specifies a list of cells from which annotated powers are to be removed. The **-all** and *cell_list* options are mutually exclusive.

DESCRIPTION

The **remove_annotated_power** command removes internal power and leakage power that were previously annotated on cells by using the **set_annotated_power** command. There is no way to remove only internal power or leakage power.

EXAMPLES

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -internal_power 1 -leakage_power 0.01 u0/*
1
pt_shell> report_annotated_power -list_annotated
****************************************
```

```
Report : annotated_power
        -list_annotated
****************************************

Annotated cell powers:
------------------------------



 1. u0/u0  (internal: 1  leakage: 0.01)

 2. u0/u1  (internal: 1  leakage: 0.01)



                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell    |        2  |        1  |        1  |
leaf cell                    |        3  |        1  |        2  |
-----------------------------+-----------+-----------+-----------+
                             |        5  |        2  |        3  |

1
pt_shell> remove_annotated_power u0/u0
1
pt_shell> report_annotated_power
****************************************
Report : annotated_power
****************************************

                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell    |        2  |        1  |        1  |
leaf cell                    |        3  |        0  |        3  |
-----------------------------+-----------+-----------+-----------+
                             |        5  |        1  |        4  |

1
```

SEE ALSO

report_annotated_power(2)
set_annotated_power(2)

remove_annotated_transition

Removes previously-annotated transition times from pins or ports in the current design.

SYNTAX

```
status remove_annotated_transition
   -all
   pin_or_port_list
```

Data Types

```
pin_or_port_list     list
```

ARGUMENTS

-all

Removes all annotated transition times in the design. You cannot use this option together with the *pin_list* option.

pin_or_port_list

Removes annotated transition times from the specified list of pins or ports. You cannot use this option together with the **-all** option.

DESCRIPTION

The **remove_annotated_transition** command removes transition times previously annotated on pins or ports from the current design using the **set_annotated_transition** command.

To view current settings, use the **report_timing** command.

EXAMPLES

The following example removes all annotated pin and port transition times from the current design.

```
pt_shell> remove_annotated_transition -all
1
```

The following example removes annotated transition time from the input pin **A** of cell "U1/U2/U3".

```
pt_shell> remove_annotated_transition [get_pins U1/U2/U3/A]
1
```

SEE ALSO

```
remove_annotated_delay(2)
report_timing(2)
reset_design(2)
set_annotated_delay(2)
set_annotated_transition(2)
```

remove_buffer

Removes specified buffers from the current design.

SYNTAX

```
int remove_buffer
   cell_list
```

Data Types

```
cell_list      list
```

ARGUMENTS

cell_list

Specifies a list of buffer cells to be removed.

DESCRIPTION

The **remove_buffer** command is used to remove buffer cells from the netlist. A buffer is one or more cells that conforms to the rules the **insert_buffer** command uses to create buffers. Like all other netlist editing commands, for the **remove_buffer** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **remove_buffer** command returns a 1 if successful and a 0 if unsuccessful.

If the input and output net of the buffer has parasitics, they are stitched into the resulting net, by default.

Similar to the **insert_buffer** command, the arguments for the **remove_buffer** command can be grouped. The basic interpretation of the *cell_list* is

```
(one non-inverting buffer | two cascaded inverting buffers)+
```

To determine whether a removable buffer configuration exists, the **remove_buffer** command applies the following rules to the cells in the *cell_list*:

- Each cell must be a buffer as defined by the **insert_buffer** command
- The input pin of each cell must be connected to a net
- For multioutput cells, only one output pin can be connected
- There can be no standalone inverting cells

- The output net cannot be multidriven
- Inverter pairs must be connected together and the intermediate net can be connected only to the two inverters

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

The following diagram contains inverters U1 and U2:

```
e1/Z -- old_net -- U1 -- net1 --- U2 -- net2 --- e3/A
```

The following example removes U1 and U2.

```
pt_shell> remove_buffer {U1 U2}
Removed buffer 'U1', 'U2'

1
```

After removing the buffer, the configuration is as follows:

```
e1/Z -- old_net --- e3/A
```

SEE ALSO

```
insert_buffer(2)
size_cell(2)
swap_cell(2)
write_changes(2)
```

remove_capacitance

Removes capacitance on nets or ports.

SYNTAX

```
status remove_capacitance
  net_or_port_list
```

Data Types

```
net_or_port_list      list
```

ARGUMENTS

net_or_port_list

Specifies a list of ports and nets in the current design, whose capacitances are removed.

DESCRIPTION

Specifies that the lumped capacitance annotated on the list of nets or ports must be removed. PrimeTime reverts to using the capacitance from detailed parasitics (set using the **read_parasitics** command), if they exist. If not, the internally estimated net capacitance is used.

Annotated capacitances are also removed on the full design by doing a *reset_design*.

EXAMPLES

The following example removes the annotated capacitance on net w23.

```
pt_shell> remove_capacitance [get_nets "w23"]
```

The following example removes the annotated capacitance on all ports that name match "P2*".

```
pt_shell> remove_capacitance [get_ports "P2*"]
```

SEE ALSO

```
all_outputs(2)
current_design(2)
set_load(2)
report_net(2)
report_port(2)
reset_design(2)
set_drive(2)
```

remove_case_analysis

Removes the case analysis value on input.

SYNTAX

```
status remove_case_analysis
  [-all]
  port_or_pin_list
```

Data Types

```
port_or_pin_list     list
```

ARGUMENTS

-all

Removes all logic constant values from the current design. Using this option overrides any port or pin specifications.

port_or_pin_list

Removes case analysis from the specified ports or pins.

DESCRIPTION

This command removes previously specified entries of case analysis on ports or pins. You must specify either the *port_or_pin_list* or **-all** option.

To specify case analysis, run the **set_case_analysis** command.

EXAMPLES

The following example specifies that ports in0 and in2 are set to the constant logic value 0:

```
pt_shell> set_case_analysis 0 {in0 in2}
pt_shell> report_case_analysis

****************************************
```

```
Report : case_analysis
 ...
****************************************

Pin name                 Case analysis value
----------------------------------------------
in2                      0
in0                      0
```

The following example removes the case analysis entry on port in2:

```
pt_shell> remove_case_analysis {in2}
pt_shell> report_case_analysis

****************************************
Report : case_analysis
 ...
****************************************

Pin name                 Case analysis value
----------------------------------------------
in0                      0
```

SEE ALSO

report_case_analysis(2)
set_case_analysis(2)

remove_cell

Removes cells from the current design.

SYNTAX

```
int remove_cell
   -all
   cell_list
```

Data Types

```
cell_list      list
```

ARGUMENTS

-all

Indicates that all cells are to be removed from the current design. The *-all* and *cell_list* options are mutually exclusive; you can specify only one.

Note: Using the *-all* option at the top of the design effectively removes the entire design.

cell_list

Specifies a list of cells to be removed from the current design. The *-all* and *cell_list* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **remove_cell** command removes cells from the current design. Like all other netlist editing commands, for the **remove_cell** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **remove_cell** command returns a 1 if successful and a 0 if unsuccessful.

Each cell specified must be in scope; that is, at or below the current instance. The **remove_cell** command can be used on leaf cells or hierarchical cells. Duplicate arguments of the **remove_cell** command are ignored with a warning. As the **remove_cell** command progresses, it echoes the cells passed to it. However, only the cells that were actually deleted are logged to the change list. For more information, see the EXAMPLES section.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The

script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following example, the first command shows the cells within u1 and u2. The second command removes what appears to be four cells; in fact, more than four cells are deleted. Because u1 and u2 each contain three cells, a total of eight cells are actually deleted. Note also that the change list has only three entries. Because u1/u2 is contained within u1 and is removed after u1, u1/u2 is dropped from the change list.

```
pt_shell> get_cells {u1/* u2/*}
{"u1/u1", "u1/u2", "u1/u3", "u2/u1", "u2/u2", "u2/u3"}
pt_shell> remove_cell {u1 u1/u2 u2/u3 u2}
Information: Removed cell 'u1'. (NED-015)
Information: Removed cell 'u1/u2'. (NED-015)
Information: Removed cell 'u2/u3'. (NED-015)
Information: Removed cell 'u2'. (NED-015)
1
pt_shell> write_changes
#####################################
# Change list, formatted for PrimeTime
```

```
#######################################
current_instance
remove_cell {u1}
current_instance
current_instance u2
remove_cell {u3}
current_instance
remove_cell {u2}
1
```

SEE ALSO

```
size_cell(2)
swap_cell(2)
write_changes(2)
```

remove_clock

Removes one or more clocks from the current design.

SYNTAX

```
string remove_clock
   -all
   clock_list
```

Data Types

```
clock_list     list
```

ARGUMENTS

-all

Specifies to remove all clocks in the current design.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names.

DESCRIPTION

Removes the specified clock objects from the current design. You must specify either the *clock_list* or *-all* options.

If a removed clock is the only member of a path group, the path group is also removed. For information about path groups, refer to the **group_path** command man page. To list all clock sources in the design, use the **report_clock** command.

All arrival and required times specified by the **set_input_delay** and **set_output_delay** commands relative to the clock that is being removed are also removed.

EXAMPLES

The following example removes clock CLK1.

```
pt_shell> remove_clock CLK1
```

The following example removes all clocks from the current design.

```
pt_shell> remove_clock -all
```

SEE ALSO

```
create_clock(2)
current_design(2)
get_clocks(2)
group_path(2)
report_clock(2)
set_input_delay(2)
set_output_delay(2)
reset_design(2)
```

remove_clock_gating_check

Removes clock-gating checks.

SYNTAX

```
status remove_clock_gating_check
   [-setup]
   [-hold]
   [-rise]
   [-fall]
   [-high]
   [-low]
   [object_list]
```

Data Types

```
object_list       list
```

ARGUMENTS

-setup

Removes the clock-gating constraint on the setup time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-hold

Removes the clock-gating constraint on the hold time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-rise

Removes the clock-gating constraint on the rising delays only. If you do not specify either the **-rise** or **-fall** option, constraints on both rising and falling delays are removed.

-fall

Removes the clock-gating constraint on the falling delays only. If you do not specify either the **-rise** nor **-fall** option, constraints on both rising and falling delays are removed.

-high

Removes the high specification from the object list, previously set up by the **set_clock_gating_check** command. This option has to be either high or low.

-low

Removes the low specification from the object list, previously set up by the **set_clock_gating_check**

command. This option has to be either high or low.

object_list

Specifies a list of objects in the current design from which to remove the clock gating check. The objects can be clocks, pins, or cells. If you specify a cell, all input pins of that cell are affected. If you do not specify any objects, the clock-gating check is removed from the current design.

DESCRIPTION

The **remove_clock_gating_check** command removes clock gating checks for design objects set by the **set_clock_gating_check** command.

An alternative way to remove information set by the **set_clock_gating_check** command is to use the **reset_design** command.

EXAMPLES

The following example removes the setup requirement (for rising and falling delays) on all gates in the clock network involved with clock CK1 path.

```
pt_shell> remove_clock_gating_check -setup [get_clocks CK1]
```

The following example removes the hold requirement on the rising delay of gate and1.

```
pt_shell> remove_clock_gating_check -hold -rise [get_cells and1]
```

SEE ALSO

```
remove_disable_clock_gating_check(2)
report_constraint(2)
reset_design(2)
set_clock_gating_check(2)
set_disable_clock_gating_check(2)
```

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

SYNTAX

```
status remove_clock_groups
   [-logically_exclusive]
   [-physically_exclusive]
   [-exclusive]
   [-asynchronous]
   [-name name_list]
   [-all]
```

Data Types

```
name_list     list
```

ARGUMENTS

-logically_exclusive

Specifies that the groups set for logically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-physically_exclusive

Specifies that the groups set for physically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-exclusive

Specifies that the groups set for logically exclusive clocks are to be removed.

-asynchronous

Specifies that groups set for asynchronous clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive**, and **-asynchronous** options are mutually exclusive; you must choose only one.

-name *name_list*

Specifies a list of clock groups to be removed, which matches the groups in the given names. These clock groups are predefined by the **set_clock_groups** command. The **-name** and **-all** options are mutually exclusive.

-all

Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The **-name** and **-all** options are mutually exclusive.

DESCRIPTION

This command removes specific exclusive or asynchronous clock groups from the current design. You must specify either the **-exclusive** or **-asynchronous** option. You must specify either the *name_list* or **-all** option.

To list the existing clock groups, run the **report_clock** command with the **-groups** option.

EXAMPLES

The following example removes exclusive clock groups named mux.

```
pt_shell> remove_clock_group -exclusive -name mux
```

The following example removes all asynchronous clock groups from the current design.

```
pt_shell> remove_clock_group -asynchronous -all
```

SEE ALSO

report_clock(2)
set_clock_groups(2)

remove_clock_latency

Removes clock latency information from specified objects.

SYNTAX

```
string remove_clock_latency
   [-source]
   [-clock clock_list]
   object_list
```

Data Types

```
clock_list        list
object_list       list
```

ARGUMENTS

-source

Specifies that clock source latency should be removed.

-clock *clock_list*

Removes any network latency defined on the pin/port objects in the *object_list* that refers the clocks in the *clock_list* option from the design. If the *-clock* option is supplied when the *object_list* refers to clock objects, a warning is issued that the option is not relevant in this case and the command proceeds as if the *-clock* was not provided. This option does not remove a more general latency setting without any specific clock.

object_list

Provides a list of clocks, ports, or pins.

DESCRIPTION

Removes clock latency information from specified objects. It removes the user-specified clock network or source latency information from specified objects. If a dynamic component of clock source latency has been specified this is also removed. Clock Network latency is the time it takes for a clock signal on the design to propagate from the clock definition point to a register clock pin. Clock Source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. Clock network and source latency information is set on objects using the **set_clock_latency** command. For more information, see the **set_clock_latency** man page.

To report clock network and source latency information, use the **report_clock** command with the *-skew* option.

EXAMPLES

The following example removes clock latency information from the **CLK1** clock.

```
pt_shell> remove_clock_latency [get_clocks CLK1]
```

The following example removes clock source latency information from the **CLK1** clock.

```
pt_shell> remove_clock_latency -source \
[get_clocks CLK1]
```

SEE ALSO

```
create_clock(2)
remove_clock(2)
remove_clock_uncertainty(2)
report_clock(2)
set_clock_latency(2)
set_clock_uncertainty(2)
```

remove_clock_sense

The **remove_clock_sense** command is deprecated. Use the **remove_sense** command instead.

SEE ALSO

remove_sense(2)
set_sense(2)

remove_clock_transition

Removes clock transition time information.

SYNTAX

```
string remove_clock_transition
   clock_list
```

Data Types

clock_list	list

ARGUMENTS

clock_list

Specifies a list of clocks for which to remove clock transition time.

DESCRIPTION

Removes clock transition information specified by the **set_clock_transition** command. To list all the set clock transition values, use the **report_clock** command with the *-skew* option.

EXAMPLES

The following example removes clock transition information from all clocks in the current design.

```
pt_shell> remove_clock_transition [all_clocks]
```

SEE ALSO

```
all_clocks(2)
report_clock(2)
set_clock_transition(2)
create_clock(2)
```

```
set_clock_latency(2)
set_clock_uncertainty(2)
```

remove_clock_uncertainty

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command.

SYNTAX

```
string remove_clock_uncertainty
   [object_list
    -from from_clock
      | -rise_from rise_from_clock
      | -fall_from fall_from_clock
    -to to_clock
      | -rise_to rise_to_clock
      | -fall_to fall_to_clock]
   [-rise]
   [-fall]
   [-setup]
   [-hold]
```

Data Types

```
object_list          list
from_clock           list
rise_from_clock      list
fall_from_clock      list
to_clock             list
rise_to_clock        list
fall_to_clock        list
```

ARGUMENTS

object_list

Specifies a list of clocks, ports, or pins from which uncertainty information is removed. You can use either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to* options or the *object_list* option, but you cannot specify both; they are mutually exclusive.

-from *from_clock*

This option specifies the source clock for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

-rise_from *rise_from_clock*

Same as the *-from* option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options. Use the *-rise_from* option instead of the obsolete *-from_edge rise* option.

-fall_from *fall_from_clock*

Same as the *-from* option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options. Use the *-fall_from* instead of the obsolete *-from_edge fall* option.

-to *to_clock*

This option specifies the destination clocks for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

-rise_to *rise_to_clock*

Same as the *-to* option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options. Use the *-rise_to* option instead of the obsolete *-to_edge rise* option.

-fall_to *fall_to_clock*

Same as the *-to* option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options. Use the *-fall_to* option instead of the obsolete *-to_edge fall* option.

-rise

Specifies that uncertainty is removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-rise_to* option instead.

-fall

Specifies that uncertainty is removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-fall_to* option instead.

-setup

Specifies that only setup check uncertainty is removed. By default, both setup and hold check uncertainties are removed.

-hold

Specifies that only hold check uncertainty is removed. By default, both setup and hold check uncertainties are removed.

DESCRIPTION

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command from clocks, ports, pins, or between specified clocks. To display clock uncertainty information, use the **report_clock** command with the *-skew* option.

EXAMPLES

The following example removes uncertainty information from a clock named *CLK* and a pin named *clk_buf/Z*.

```
pt_shell> remove_clock_uncertainty [get_clocks CLK]
pt_shell> remove_clock_uncertainty [get_pins clk_buf/Z]
```

The following example removes interclock uncertainties between the *PHI1* and *PHI2* clock domains.

```
pt_shell> remove_clock_uncertainty -from PHI1 -to PHI1
pt_shell> remove_clock_uncertainty -from PHI2 -to PHI2
pt_shell> remove_clock_uncertainty -from PHI1 -to PHI2
pt_shell> remove_clock_uncertainty -from PHI2 -to PHI1
```

SEE ALSO

```
all_clocks(2)
create_clock(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
```

remove_connection_class

Removes connection class value from ports.

SYNTAX

```
integer remove_connection_class
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Specifies ports whose connection classes are to be removed.

DESCRIPTION

Removes any values for previously set on port with the **set_connection_class** command.

EXAMPLES

The following example removes any existing connection class values for an xyz port name.

```
pt_shell> remove_connection_class xyz
```

SEE ALSO

```
set_connection_class(2)
report_constraint(2)
```

remove_context

Deletes timing context information previously created by the **characterize_context** command.

SYNTAX

```
string remove_context
   -block block_name
   [-instances cell_list]
```

Data Types

```
block_name    string
cell_list     list
```

ARGUMENTS

-block *block_name*

Specifies the name of the design subblock for which its context to be removed. This argument is required.

-instances *cell_list*

Specifies a list of one or more instance names in the current design, all of which must all be instances of the reference block specified by the **-block** *block_name* argument. If you do not use the **-instances** option, the command affects all instances of the block named by the **-block** argument.

DESCRIPTION

This command deletes the context information previously captured by the **characterize_context** command for the specified list of instances. When you need to recharacterize the context for some instances, first use the **remove_context** command to remove the previously generated context for those instances.

EXAMPLES

The following command deletes the context information for instances I1 and I2.

```
pt_shell> remove_context -instances {I1 I2} -block B
```

SEE ALSO

```
characterize_context(2)
write_context(2)
report_context(2)
```

remove_coupling_separation

Removes the constraints set by the **set_coupling_separation** command.

SYNTAX

```
status remove_coupling_separation
   [-pairwise pnets]
   -all
   nets
```

Data Types

```
pnets            list
nets             list
```

ARGUMENTS

-pairwise *pnets*

Specifies that all coupling capacitances between only *pnets* and *nets* are excluded. The **-pairwise** option can only be used to remove separation constraints applied on the nets using the **set_coupling_separation** command with the **-pairwise** option. Any coupling separation applied on the nets globally without the **-pairwise** option cannot be reset by this option.

-all

Resets coupling separation constraints on all nets.

nets

Specifies a list of nets from which separation constraints are to be removed.

DESCRIPTION

This command removes the constraints set by the **set_coupling_separation** command.

This command restores any prior settings on *pnets* or *nets* from the **set_si_delay_analysis** or **set_si_noise_analysis** command.

Instances of this command are recorded for output by the **write_changes** command. This feature allows you to communicate the separation constraint to other implementation tools along with netlist changes.

To verify the result of this command, run the **report_si_delay_analysis** or **report_si_noise_analysis**

command with the **-coupling_separated** option.

EXAMPLES

In the following example, all the nets named CLK_NET_* are returned to their original state in crosstalk analysis.

```
pt_shell> set_coupling_separation [get_nets CLK_NET*]
1
pt_shell> remove_coupling_separation [get_nets CLK_NET*]
1
```

In the following example, the separation is applied pairwise on the nets and is removed by using the **-pairwise** option.

```
pt_shell> set_coupling_separation -pairwise [get_nets {n1 n2}] [get_nets n3]
1
pt_shell> remove_coupling_separation -pairwise [get_nets {n1 n2}] [get_nets n3]
1
```

However if the separation is applied globally on the net for all its coupled nets, it cannot be removed by using the **-pairwise** option.

```
pt_shell> set_coupling_separation [get_nets {n4}]
1
pt_shell> remove_coupling_separation [get_nets {n4}] -pairwise [get_nets n3]
Warning: Cannot remove an effect that was not set on net(s) n4 n3. (XTALK-107)
1
```

The coupling separation can be applied on the net pairwise with all its coupled nets in the following way so that pairwise exclusions can be applied on the net with any of its coupled nets.

```
pt_shell> set_coupling_separation [get_nets VIC_NET] -pairwise
[get_attribute -class net [get_nets VIC_NET] aggressors]
1

pt_shell> remove_coupling_separation [get_nets VIC_NET] -pairwise
[get_nets AGG_NET*]
1

pt_shell> remove_coupling_separation -all
1
```

SEE ALSO

report_si_delay_analysis(2)
report_si_noise_analysis(2)
set_coupling_separation(2)
write_changes(2)

remove_data_check

Removes specified data-to-data checks previously set by the **set_data_check** command.

SYNTAX

```
string remove_data_check
   [-from from_object
      | -rise_from from_object
      | -fall_from from_object]
   [-to to_object
      | -rise_to to_object
      | -fall_to to_object]
   [-setup | -hold]
   [-clock clock_object]
```

Data Types

```
from_object       collection
to_object         collection
clock_object      collection
```

ARGUMENTS

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check is removed. Both rising and falling checks are removed. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

-rise_from *from_object*

Similar to the *-from* option, but applies only to rising delays at the related pin. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

-fall_from *from_object*

Similar to the *-from* option, but applies only to falling delays at the related pin. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

-to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check is created. Both rising and falling checks are removed. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

-rise_to *to_object*

Similar to the *-to* option, but applies only to rising delays at the constrained pin. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

-fall_to *to_object*

Similar to the *-to* option, but applies only to falling delays at the constrained pin. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

-setup

Indicates that only the setup data check is removed. If neither the *-setup* or *-hold* option is specified, both setup and hold checks are removed.

-hold

Indicates that only the hold data check is removed. If neither the *-setup* or *-hold* is specified, both setup and hold checks are removed.

-clock *clock_object*

Indicates that the data check for the specified clock at the related pin is removed. This option applies only if a previous **set_data_check** command used the *-clock* option to specify the same clock; otherwise, this option is ignored.

DESCRIPTION

The **remove_data_check** command specifies data-to-data checks are removed between the from object and to object for the specified options. These checks were previously set using the **set_data_check** command.

EXAMPLES

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
pt_shell> remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data check between and1/B to and1/A with respect to the rising edge of signals at and1/B, coming from startpoints triggered with clock CK1, falling edge of signals at and1/A.

```
pt_shell> remove_data_check -rise_from and1/B -fall_to and1/A -setup -clock [get_clock CK1]
```

SEE ALSO

report_timing(2)
set_data_check(2)
update_timing(2)

remove_design_mode

Removes specified design modes and/or cell mode and path mappings to design modes.

SYNTAX

```
status remove_design_mode
   [-from from_pin_list]
   [-to to_pin_list]
   [-through through_pin_list]
   mode_list
   [cell_mode_list]
   [instance_list]
```

Data Types

```
from_pin_list        list
to_pin_list          list
through_pin_list     list
mode_list            list
cell_mode_list       list
instance_list        list
```

ARGUMENTS

-from *from_pin_list*

Specifies a timing path, from a given pin of the design, that is active only for the specified design mode. The given paths are inactive for other design modes. These paths are unmapped from the specified design mode and any previous settings on the path due to the design mode are removed.

-to *to_pin_list*

Specifies a timing path to a given pin of the design, that is active only for the specified design mode. The given paths are inactive for other design modes. These paths are unmapped from the specified design mode and any previous settings on the path due to the design mode are removed.

-through *through_pin_list*

Lists the pins, ports, or nets through which the paths pass. The paths become active for the specified mode. You can specify the **-through** option more than one time in one command invocation. Nets are interpreted to imply the leaf-level driver pins. These paths are unmapped from the **design_mode and** command and any previous settings on the path due to the design mode are reset.

mode_list

Specifies a list of design modes to be removed. Design modes on the list must have been previously created using the **define_design_mode_group** command.

cell_mode_list

Specifies a list of cell modes to be unmapped from the specified design mode. Any previous settings on any cell mode due to the design mode are reset.

instance_list

Specifies a list of cells in which the specified modes are unmapped from the design mode. This list must be accompanied by the **cell_mode_list** option.

DESCRIPTION

The **remove_design_mode** command removes specified design modes previously created by the **define_design_mode_group** command or removes mappings for specified design modes created by the **map_design_mode** command.

A design mode can be removed from a design mode group using the **remove_design_mode** command. When a design mode is removed from a design mode group, all previous settings on cell modes and paths are reset. Also, if this design mode was the active design mode, then all other design modes in the design mode group are reset to default.

A path can be unmapped from a design mode using the **remove_design_mode -from [from_pin_list] -to [to_pin_list] -through [through_pin_list]** command. The pin specification must exactly match the pin specification set using a previous **map_design_mode** command. When a path is unmapped from a design mode, all paths are reset if they have been previously set false by that design mode. When a cell mode is unmapped from a design mode, all cell modes are reset if they have been previously set by that design mode.

To see a report of the design modes specified for the current design, use the **report_mode** command. The report also shows the cell modes and paths mapped to each design mode.

EXAMPLES

The following example defines two design modes, DM1 and DM2, maps the cell mode READ to design mode DM1 for instance Uram1, then removes the design mode DM1. Removing DM1 also removes the mapping to the active cell mode READ.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode READ Uram1 DM1
pt_shell> remove_design_mode DM1
```

SEE ALSO

```
define_design_mode_group(2)
map_design_mode(2)
report_mode(2)
reset_mode(2)
set_mode(2)
```

remove_disable_clock_gating_check

Restores clock gating checks, previously disabled by the **set_disable_clock_gating_check** command, for specified cells and pins.

SYNTAX

```
string remove_disable_clock_gating_check
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Specifies a list of cells, pins, lib-pins or lib-cells for which previously-disabled clock gating checks are to be restored.

DESCRIPTION

Restores timing clock gating checks previously disabled with the **set_disable_clock_gating_check** command.

EXAMPLES

The following example restores disabled clock gating checks for the object an1.

```
pt_shell> remove_disable_clock_gating_check an1/A
```

The following example restores all disabled gating checks on the cell U1/or2.

```
pt_shell> remove_disable_clock_gating_check U1/or2
```

The following example restores all disabled gating checks on the lib-cell class/AND1.

```
pt_shell> remove_disable_clock_gating_check [get_lib_cells {"class/AND1"}]
```

The following example restores all disabled gating checks on the lib-pin class/AND1/A.

```
pt_shell> remove_disable_clock_gating_check [get_lib_pins {"class/AND1/A"}]
```

SEE ALSO

set_disable_clock_gating_check(2)

remove_disable_timing

Enables the previously disabled timing arcs.

SYNTAX

```
status remove_disable_timing
   [-from from_pin_name]
   [-to to_pin_name]
   object_list
```

Data Types

```
from_pin_name      string
to_pin_name        string
object_list        list
```

ARGUMENTS

-from *from_pin_name*

Specifies that only the arcs from this pin on the specified cell or library cell are to be disabled. When used, the **-from** and **-to** options must be specified together.

-to *to_pin_name*

Specifies that only the arcs to this pin on the specified cell or library cell are to be disabled. When used, the **-from** and **-to** options must be specified together.

object_list

Specifies a list of cells, pins, ports, library cells, library cell pins, cell timing arcs, or library timing arcs. If the **-from** and **-to** options are specified, the *object_list* argument must contain only cells or library cells.

DESCRIPTION

Restore timing arcs previously disabled with the **set_disable_timing** command. When the **-from** and **-to** option pins are specified, all arcs between these two pins on the cell or library cell are restored.

To list the timing arcs for a library cell, use the **report_lib -timing_arcs** command.

EXAMPLES

The following example restores disabled setup and hold arcs between pins CLK and TE on library cell SFF1.

```
pt_shell> remove_disable_timing -from CLK -to TE tech_lib/SFF1
```

The following example restores all disabled cell arcs from or to pin A on cell U1/U2.

```
pt_shell> remove_disable_timing U1/U2/A
```

SEE ALSO

```
report_disable_timing(2)
report_lib(2)
set_disable_timing(2)
```

remove_drive_resistance

Removes drive resistance for input or inout ports.

SYNTAX

```
string remove_drive_resistance
   port_list
```

Data Types

```
port_list        list
```

ARGUMENTS

port_list

Specifies a list of input or inout port names of the current design from which the drive values are to be removed.

DESCRIPTION

The **remove_drive_resistance** command removes the drive resistance value from one or more input or inout ports. This is equivalent to using the **set_drive_resistance 0** command. In fact, that is how this command works. So, if output ports are passed into the **remove_drive_resistance** command, a DES-003 message is issued, indicating that the **set_drive_resistance** command could not be performed on those ports.

EXAMPLES

The following command shows the difference in the output report after using the **remove_drive_resistance** command:

```
pt_shell> set_drive_resistance 5 [get_ports {A B C}]
1
pt_shell> report_port -drive {A B C}

****************************************
Report : port
         -drive
Design : counter
****************************************
```

```
                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
-------------------------------------------------------------
A             5.00      5.00      --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --

1
pt_shell> remove_drive_resistance A
1
pt_shell> report_port -drive {A B C}

****************************************
Report : port
        -drive
Design : counter
****************************************


                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
-------------------------------------------------------------
A              --        --       --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --

1
```

SEE ALSO

report_port(2)
set_load(2)
set_drive_resistance(2)

remove_driving_cell

Removes port driving cell information.

SYNTAX

```
string remove_driving_cell
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-clock clock_name]
   [-clock_fall]
   port_list
```

Data Types

```
clock_name      string
port_list       list
```

ARGUMENTS

-rise

Removes rise driving cell information.

-fall

Removes fall driving cell information.

-min

Removes min driving cell information.

-max

Removes max driving cell information.

-clock *clock_name*

Removes the driving cell set relative to the specified clock.

-clock_fall

Removes the driving cell relative to the falling edge of the clock. The default is the rising edge.

port_list

Provides a list of input or output ports.

DESCRIPTION

Removes driving cell information from the specified ports. The driving cell information is specified with the **set_driving_cell** command. The default is to remove all the *-rise* and *-fall* option information.

To see port drive information, use the **report_port -drive** command.

EXAMPLE

The following example removes all driving cell information for port IN2.

```
pt_shell> remove_driving_cell IN2
```

SEE ALSO

```
report_port(2)
set_driving_cell(2)
```

remove_fanout_load

Removes fanout load information from output ports in the current design.

SYNTAX

```
status remove_fanout_load
   port_list
```

Data Types

```
port_list         list
```

ARGUMENTS

port_list

Specifies a list of output ports.

DESCRIPTION

Removes fanout load information specified by the **set_fanout_load** command. Fanout load for a net is the sum of the **fanout_load** attributes for all input pins and output ports connected to the net. Output pins can have maximum fanout limits, specified in the library or with the **set_max_fanout** command. The **fanout_load** is used when checking max_fanout design values.

Use the **report_constraint** command with the **-max_fanout** option to show maximum fanout constraint evaluations. Use the **report_port** command with the **-design_rule** option to show port fanout load values.

EXAMPLES

The following example removes the fanout load on ports matching "OUT*".

```
pt_shell> remove_fanout_load "OUT*"
```

SEE ALSO

```
report_constraint(2)
report_port(2)
set_fanout_load(2)
set_max_fanout(2)
```

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection remove_from_collection
   [-intersect]
   collection1
   object_spec
```

Data Types

collection1	collection
object_spec	list

ARGUMENTS

-intersect

Removes objects from collection1 not found in object_spec. Without this option, removes objects from collection1 that are found in object_spec.

collection1

Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec

Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

DESCRIPTION

The **remove_from_collection** command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, any element of the *object_spec* that is not a collection is ignored.

If the *-intersect* option is not specified, which is the default mode, and if nothing matches the *object_spec*, the resulting collection is a copy of the base collection. If everything in the *collection1* option matches the *object_spec*,

the result is the empty collection. With the *-intersect* option the results are reversed.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime gets all input ports except "CLOCK".

```
pt_shell> set cPorts [remove_from_collection [all_inputs] CLOCK]
{"in1", "in2"}
```

SEE ALSO

add_to_collection(2)
collections(2)

remove_generated_clock

Removes generated clock objects from the current design.

SYNTAX

```
status remove_generated_clock
   -all
   clock_list
```

Data Types

```
clock_list        list
```

ARGUMENTS

-all

Indicates that all generated clocks are to be removed.

clock_list

Specifies a list of names of generated clocks to be removed.

DESCRIPTION

This command removes from the current design generated clocks previously created using the **create_generated_clock** command. All attributes set on generated clocks such as, *false_paths* and *multicycle_paths*, are also removed.

To display information about clocks and generated clocks in the design, use the **report_clock** command.

EXAMPLES

The following example removes the generated clock GEN1 from the design.

```
pt_shell> remove_generated_clock GEN1
```

The following example removes all generated clocks from the design.

```
pt_shell> remove_generated_clock -all
```

SEE ALSO

```
create_generated_clock(2)
report_clock(2)
```

remove_host_options

Removes hosts options set using the **set_host_options** command.

SYNTAX

```
int remove_host_options
   [host_option_names]
```

Data Types

host_option_names	list

ARGUMENTS

host_option_names

This option specifies a list of named host options to remove.

DESCRIPTION

Given a list of host option names, removes those host options. The host option names specifed must have been previously set using the **set_host_options** command. If no host option names are specified, all host options are removed. Regardless of whether host option names are specified or not, all processes are shutdown when any host options are removed.

In a DMSA master process, if a host option is removed then it will also be removed from the affinity specificaiton of all scenarios. If all host options are removed then all scenairo affinities will be removed. Where scenarios specify an affinity the removal of host opitons will reduce the set of workers upon which the scenario can run.

EXAMPLES

In the following example, the master process is running on the host named ptnlm15 using a 64-bit binary and specifies several different sets of host options.

The first host options, named "my_opts1", specifies

- to use one process on the same host as the master
- no upper limit on the number of cores to use so the default applies.

The second host options, named "my_opts2", specifies

- to use one process on the same host as the master, explicitly mentioning the master host by name
- to use a maximum of two cores.

The third host options, named "my_opts3", specifies

- to use one process on the host named ptnlm11, since no submit command is specified, "rsh" will be used to connect to ptnlm11
- to use a maximum of three cores.

The fourth host options named "my_opts4", specifies

- to use one process on the host named ptnlm12 connecting to ptnlm12 using the submit command "/usr/bin/rsh"
- no upper limit on the numebr of cores to use so the default applies.

The fifth host options named "my_opts5", specifies

- to use one process on an LSF farm, using the "bsub" command found from the environment path to submit the job, requiring 1GB of memory and two slots per compute server. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use with the **-n 2 -R span\[ptile=2\]** options.
- to use the terminate command "bkill" found from the environment path for the termination of jobs that do not come online.
- to use a maximum of two cores.

The sixth host options named "my_opts6", specifies

- to use one process on a Grid farm, using the "qsub" command found from the environment path to submit the job, using the project named "bnormal", requiring 1GB of memory and four slots per compute server. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use by targeting the parallel environment "mt" requesting 4 slots with the "-pe mt 4" options.
- to use the terminate command "qdel" found from the environment path for the termination of jobs that do not come online.
- to use a maximum of four cores.

Note: After the **remove_host_options** command has been called all processes are shutdown and all the host options are removed.

Call the **set_host_options** command to specify the host options.

```
pt_shell> set_host_options -name my_opts1 -num_processes 1
1
pt_shell> set_host_options -name my_opts2 -num_processes 1 \
          ptnlm15 -max_cores 2
1
pt_shell> set_host_options -name my_opts3 -num_processes 1 \
          ptnlm11 -max_cores 3
Warning: no submit command specified to access remote host
 'ptnlm10'. The 'rsh' command will be used. (CMCR-004)
1
pt_shell> set_host_options -name my_opts4 -num_processes 1 \
          -submit_command "/usr/bin/rsh" ptnlm12
1
pt_shell> set_host_options -name my_opts5 -num_processes 1 \
          -submit_command "[sh which bsub] -n 2 -R rusage\[mem=1000\] \
          -R span\[ptile=2\]" -terminate_command "[sh which bkill]" \
          -max_cores 2
1
pt_shell> set_host_options -name my_opts6 -num_processes 1 \
          -submit_command "[sh which qsub] -P bnormal -pe mt 4 \
```

```
          -l mem_free=1G" -terminate_command "[sh which qdel]" \
          -max_cores 4
1
```

Call the **start_hosts** command to start the slave processes. Notice that each instance launched is assigned a number e.g. 1] Launched …

```
pt_shell> start_hosts
Launching 6 Distributed Workers

   1] Launched : /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 4

   2] Launched : ptnlm15 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 2

   3] Launched : rsh ptnlm11 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 3

   4] Launched : /usr/bin/rsh ptnlm12 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 4

   5] Launched : /lsf/bin/bsub -n 2 -R rusage[mem=1000] -R span[ptile=2] \
                 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 2

   6] Launched : /remote/sge2/default/bin/lx-amd64/qsub -P bnormal \
                 -pe mt 4 -l mem_free=1G /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 4

  ---------------------------------------------------------------------------
  Distributed farm creation timeout : 21600 seconds
  Waiting for   6 (of   6) distributed hosts (Tue Nov 25 04:39:34 2014)
  Waiting for   2 (of   6) distributed hosts (Tue Nov 25 04:39:49 2014)
  Waiting for   0 (of   6) distributed hosts (Tue Nov 25 04:39:59 2014)
  ---------------------------------------------------------------------------

1
```

Call the **report_host_usage** command after starting the hosts to report the host options specifed by the **set_host_options** command, as well as the real time data associated with the processes.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 04:59:46 2014
****************************************

  Options Name          Host Name              Num Processes
  ---------------------------------------------------------------------
  my_opts1              **localhost**          1
  my_opts2              **ptnlm15**            1
  my_opts3              ptnlm11                1
  my_opts4              ptnlm12                1
  my_opts5              >>farm<<               1
  my_opts6              >>farm<<               1

  Options Name     #    Host Name    Job ID    Process ID    Status
  ---------------------------------------------------------------------
  my_opts1         1                 27337     27337         ONLINE
  my_opts2         2    ptnlm15      27378     27378         ONLINE
  my_opts3         3    ptnlm11      23716     23716         ONLINE
```

```
  my_opts4         4    ptnlm12      13059     13059         ONLINE
  my_opts5         5    maddog136    312487    21901         ONLINE
  my_opts6         6    peemt32      2165476   5293          ONLINE

  Usage limits (cores)

  Options Name     #                                          Effective
  ---------------------------------------------------------------------
  (local process)  -                                          4
  my_opts1         1                                          -
  my_opts2         2                                          -
  my_opts3         3                                          -
  my_opts4         4                                          -
  my_opts5         5                                          -
  my_opts6         6                                          -
  ---------------------------------------------------------------------
  Total                                                       4

  Memory usage

  Options Name     #                            Memory (MB)
  ---------------------------------------------------------------------
  (local process)  -                            445.43
  my_opts1         1                            0.00
  my_opts2         2                            0.00
  my_opts3         3                            0.00
  my_opts4         4                            0.00
  my_opts5         5                            0.00
  my_opts6         6                            0.00

  Performance

  Options Name     #    CPU Time (s)            Elapsed Time (s)
  ---------------------------------------------------------------------
  (local process)  -    2                       1023
  my_opts1         1    0                       0
  my_opts2         2    0                       0
  my_opts3         3    0                       0
  my_opts4         4    0                       0
  my_opts5         5    0                       0
  my_opts6         6    0                       0

1
```

Call the **remove_host_options** command to remove all host options. Note: After the **remove_host_options** command has been called all processes are shutdown and all the host options are removed.

```
pt_shell> remove_host_options
Shutting down all slaves processes ...
 Shutdown Process 1
 Shutdown Process 2
 Shutdown Process 3
 Shutdown Process 4
 Shutdown Process 5
 Shutdown Process 6
1
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 06:58:46 2014
****************************************
```

```
    Usage limits (cores)

    Options Name     #                                          Effective
    ---------------------------------------------------------------------
    (local process)  -                                          4
    ---------------------------------------------------------------------
    Total                                                       4

    Memory usage

    Options Name     #                            Memory (MB)
    ---------------------------------------------------------------------
    (local process)  -                            330.80

    Performance

    Options Name     #    CPU Time (s)            Elapsed Time (s)
    ---------------------------------------------------------------------
    (local process)  -    2                       52

  1
```

SEE ALSO

set_host_options(2)
report_host_usage(2)
start_hosts(2)
stop_hosts(2)

remove_ideal_latency

Removes ideal latency values from the specified objects.

SYNTAX

```
int remove_ideal_latency
   [-rise]
   [-fall]
   [-min]
   [-max]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-rise

Specifies that only rise ideal latency should be removed. By default, both rise and fall ideal latency are removed.

-fall

Specifies that only fall ideal latency should be removed. By default, both rise and fall ideal latency are removed.

-min

Specifies that only minimum ideal latency should be removed. By default, both minimum and maximum ideal latency are removed.

-max

Specifies that only maximum ideal latency should be removed. By default, both minimum and maximum ideal latency are removed.

object_list

Specifies a list of ports or pins from where to remove ideal latency.

DESCRIPTION

Removes the user-specified ideal latency that was previously set by the **set_ideal_latency** command from

specified objects in the **object_list** argument.

You can remove selected portions of ideal latency by specifying applicable *-rise*, *-fall*, *-min*, and *-max* options.

To list ideal latency values, use the **report_ideal_network** command.

EXAMPLES

The following example removes ideal latency from the output pin named *Z* of cell instance *U1/U2/U3*.

```
pt_shell> remove_ideal_latency {U1/U2/U3/Z}
```

The following example removes only rise ideal latency information from a port named *A*.

```
pt_shell> remove_ideal_latency -rise {A}
```

SEE ALSO

```
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
report_timing(2)
set_ideal_network(2)
set_ideal_latency(2)
```

remove_ideal_network

Removes sources of ideal networks in the current design. Cells and nets in the transitive fanout of the specified objects are no longer treated as ideal.

SYNTAX

```
int remove_ideal_network
   object_list
```

Data Types

```
object_list         list
```

ARGUMENTS

object_list

Specifies a list of ideal sources. The source objects can be ports or pins of leaf cells at any hierarchical level of the design. If nets are specified in the *object_list* option, all of the nets' global driver ideal source pins that were set with the *-no_propagate* option are removed.

DESCRIPTION

This command removes the ideal network property from a set of ports or pins in the design that were previously marked as ideal sources using the **set_ideal_network** command. You specify only the source of the network. The ideal network is automatically updated by PrimeTime. The criteria for propagating the ideal property are detailed in the **set_ideal_network** man page.

All ideal networks are removed using the **reset_design** command.

EXAMPLES

The following example sets up an ideal network on objects in the design CLOCK_GEN and then removes part of the network:

```
pt_shell> current_design CLOCK_GEN
pt_shell> set_ideal_network {port1 port2}
pt_shell> remove_ideal_network port2
```

The following example creates an ideal network and then removes part of the ideal network.

```
pt_shell> current_design {TOP}
pt_shell> set_ideal_network {IN1 IN2}
pt_shell> remove_ideal_network {IN1}
```

The following example removes an ideal network that was set on a net.

```
pt_shell> set_ideal_network -no_propagate {netB}
Warning: Transferring ideal net attribute onto driver pin 'AN2/Z' of net 'netB'. (UITE-450)
pt_shell> remove_ideal_network {netB}
```

SEE ALSO

remove_ideal_latency(2)
remove_ideal_transition(2)
report_ideal_network(2)
reset_design(2)
set_ideal_network(2)

remove_ideal_transition

Removes ideal transition values from the specified objects.

SYNTAX

```
int remove_ideal_transition
   [-rise]
   [-fall]
   [-min]
   [-max]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-rise

Specifies that only rise ideal transition should be removed. By default, both rise and fall ideal transitions are removed.

-fall

Specifies that only fall ideal transition should be removed. By default, both rise and fall ideal transitions are removed.

-min

Specifies that only minimum ideal transition should be removed. By default, both minimum and maximum ideal transitions are removed.

-max

Specifies that only maximum ideal transition should be removed. By default, both minimum and maximum ideal transitions are removed.

object_list

Specifies a list of ports or pins from which to remove ideal transition.

DESCRIPTION

Removes the user-specified ideal transition that was previously set by the **set_ideal_transition** command from specified objects in the *object_list* option.

You can remove selected portions of ideal transition by specifying applicable *-rise*, *-fall*, *-min*, and *-max* options.

To list ideal transition values, use the **report_ideal_network** command.

EXAMPLES

The following example removes ideal transition from the output pin named *Z* of cell instance *U1/U2/U3*.

```
pt_shell> remove_ideal_transition {U1/U2/U3/Z}
```

The following example removes only rise ideal transition information from a port named *A*.

```
pt_shell> remove_ideal_transition -rise {A}
```

SEE ALSO

```
remove_ideal_latency(2)
remove_ideal_network(2)
report_ideal_network(2)
report_timing(2)
set_ideal_network(2)
set_ideal_transition(2)
```

remove_input_delay

Removes input delay information from ports or pins.

SYNTAX

```
string remove_input_delay
   [-clock clock_name]
   [-clock_fall]
   [-level_sensitive]
   [-rise]
   [-fall]
   [-max]
   [-min]
   port_pin_list
```

Data Types

```
clock_name          list
port_pin_list       list
```

ARGUMENTS

-clock *clock_name*

Relative clock; '' for no clock. Use this option to remove only input delay relative to one clock.

-clock_fall

Delay is relative to falling edge of clock.

-level_sensitive

Delay is from level-sensitive latch.

-rise

Removes rising input delay.

-fall

Removes falling input delay.

-max

Removes maximum input delay.

-min

Removes minimum input delay.

port_pin_list

Specifies a list of ports and pins.

DESCRIPTION

Removes input delay information from ports or pins in the current design. Input delay is specified with the **set_input_delay** command. To show input delays on ports, use the **report_port -input_delay** command. The default is to remove all input delay information in the *port_pin_list* option.

EXAMPLES

The following example removes all input delay from ports IN*.

```
pt_shell> remove_input_delay [get_ports IN*]
```

The following example removes input delay relative to CLK rise edge from port DATA[5].

```
pt_shell> remove_input_delay -clock CLK { DATA[5] }
```

SEE ALSO

```
report_port(2)
set_input_delay(2)
set_output_delay(2)
remove_output_delay(2)
```

remove_input_noise

Removes input noise for a library pin or port.

SYNTAX

```
int remove_input_noise
   [-above]
   [-below]
   [-low]
   [-high]
   object_list
```

Data Types

list *object_list*

ARGUMENTS

-above

Removes the input noise for above ground or power rail noise analysis region.

-below

Removes the input noise for below ground or power rail noise analysis region.

-low

Removes the input noise for ground rail noise.

-high

Removes the input noise for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

DESCRIPTION

This command removes the input noise information set by the **set_input_noise** command.

EXAMPLES

This example removes input noise information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_input_noise -above lsi_10k/IV/A
```

SEE ALSO

set_input_noise(2)

remove_integrity_result

Removes an integrity result data set from the session.

SYNTAX

```
status remove_integrity_result
   result_name
```

Data Types

```
string result_name
```

ARGUMENTS

result_name

Specifies the name of the integrity result to be removed.

DESCRIPTION

The command removes the specified integrity result from the current session. Integrity results are generated by the **check_supply_net_integrity** command and are cached to disk during the session. These integrity result disk files are automatically deleted when you exit the session.

If you execute multiple **check_supply_net_integrity** commands during the session, you might have results that are no longer useful but still show up in the reports and consume disk space. Use the **remove_integrity_result** command to remove these integrity results that are no longer needed. Use the **report_integrity_results** command to check which integrity results are still available in the current session.

EXAMPLES

The following example first sets the integrity result, named MY_INTEGRITY_RESULT_1, as current in the session. Then the command checks which integrity results are in the disk with the **report_integrity_results** command, and removes the results from the from the memory and cached disk for the current session.

```
pr_shell> current_integrity_result
MY_INTEGRITY_RESULT_1
```

```
pr_shell> report_integrity_results
****************************************
Report : integrity check results
Design : Micro
Version: E-2010.12
Date   : Mon Jan 17 17:52:41 2011
****************************************
   MY_INTEGRITY_RESULT_1
   INTEGRITY_RESULT_2
1

pr_shell> remove_integrity_result MY_INTEGRITY_RESULT_1
1

pr_shell> current_integrity_result
none

pr_shell> report_integrity_results
****************************************
Report : integrity check results
Design : Micro
Version: E-2010.12
Date   : Mon Jan 17 17:52:41 2011
****************************************
   INTEGRITY_RESULT_2
1

pr_shell> remove_integrity_result INTEGRITY_RESULT_2
1
```

SEE ALSO

current_integrity_result(2)
report_integrity_results(2)
rename_integrity_result(2)
pr_work_dir(3)

remove_intrinsic_rc

Removes the RC values from the PG pins. These RC values were previously assigned by the **set_intrinsic_rc** command.

SYNTAX

```
status remove_intrinsic_rc
   object_list
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

A list of PG pins to which to remove intrinsic resistance and capacitance. Each item in the list can be a PG name pattern or a collection.

DESCRIPTION

The **remove_intrinsic_rc** command removes the intrinsic RC from the PG pins. These RC values were previously assigned by the **set_intrinsic_rc** command. This command returns 1 if successful, and 0 if unsuccessful.

If the effect of the **set_intrinsic_rc** command was to override the intrinsic RC values calculated by the **update_currents** command, executing the **remove_intrinsic_rc** command restores the calculated intrinsic RC values.

When the command run is complete, the PG pin *intrinsic_rc* attribute for the affected PG pins reflects the calculated intrinsic RC for the PG pin if the **update_currents** command has been executed.

EXAMPLES

The following example assigns intrinsic RC values to the PG pin vdd of the cell instance U123, and removes the assigned intrinsic RC values from the PG pin.

```
pr_shell> set_intrinsic_rc {1.0 2.0 3.0 4.0} U123/vdd
```

```
pr_shell> remove_intrinsic_rc [get_pg_pins U123/vdd]
```

SEE ALSO

```
set_intrinsic_rc(2)
update_currents(2)
```

remove_lib

Removes one or more libraries from memory.

SYNTAX

```
string remove_lib
   -all
   libraries
```

Data Types

```
libraries         list
```

ARGUMENTS

-all

Removes all libraries.

libraries

Provides a list of libraries to remove.

DESCRIPTION

This command removes a list of libraries. You must specify either the *libraries* or *-all* option. For a library to be removed, none of its library cells can be instantiated in any design, nor can any environment information (such as operating conditions or wire load models) be in use from the library. If this is the case, a message is issued and you must remove the design by first using the **remove_design** command and then using the **remove_lib** command.

EXAMPLES

The following command removes all the libraries read in.

```
pt_shell> remove_lib -all
Removing library '/u/libraries/cmos1.db:cmos1'...
1
```

The following command removes a library that was part of a max/min pair created by the **set_min_library** command.

```
pt_shell> remove_lib lib_ff
Removed max/min library relationship:
  Max: /u/libraries/lib_ss.db:lib_ss
  Min: /u/libraries/lib_ff.db:lib_ff
Removing library '/u/libraries/lib_ff.db:lib_ff'...
1
```

SEE ALSO

```
list_libs(2)
list_designs(2)
remove_design(2)
set_min_library(2)
```

remove_license

Checks in a licensed feature.

SYNTAX

```
status remove_license
   [-keep num_licenses]
   feature_list
```

Data Types

num_licenses	integer
feature_list	list

ARGUMENTS

-keep *num_licenses*

Specifies the number of licenses to retain for each feature after the command has completed. If you do not use this option, the command checks in all of the features licenses. Incremental licensing must be enabled if you want to reduce number of licenses to a number larger than zero.

feature_list

Removes the licenses of the specified list of features. To specify more than one feature, enclose the list in curly braces ({ }). For a list of all features that are available at your site, see the key file.

DESCRIPTION

This command removes (checks in) the specified licensed features from the features you are currently using.

For multicore runs, where multiple licenses might be required for certain features, you can use the **-keep** option to restrict the number of licenses that are checked in. This is analogous to the **get_license -quantity** command.

Reducing the number of licenses to a number larger than zero requires incremental licensing to be enabled. Incremental licensing is enabled by default. To disable it, set the **SNPSLMD_ENABLE_INCREMENTAL_LICENSE_HANDLING** UNIX enviroment variable to **FALSE**.

To list the features that you currently have checked out, run the **list_licenses** command.

EXAMPLES

This example checks in the PrimeTime PX and PrimeTime SI licenses.

```
pt_shell> remove_license {PrimeTime-PX PrimeTime-SI}
Checked in license 'PrimeTime-PX' (PT-020)
Checked in license 'PrimeTime-SI' (PT-020)
1
```

SEE ALSO

```
get_license(2)
license_users(2)
list_licenses(2)
```

remove_max_area

Removes the **max_area** attribute from the current design.

SYNTAX

```
int remove_max_area
```

ARGUMENTS

None.

DESCRIPTION

The **remove_max_area** command removes the **max_area** attribute from the current design. This attribute represents the target area of the design.

Use the **set_max_area** command to set the **max_area** attribute on the current design.

Use the **report_constraint** command to show area cost of the design.

EXAMPLES

The following example removes the target area from the current design.

```
pt_shell> remove_max_area
1
pt_shell> get_attribute [get_design [current_design]] max_area
Warning: Attribute 'max_area' does not exist on design 'TOP' (ATTR-3)
```

SEE ALSO

```
current_design(2)
get_attribute(2)
set_max_area(2)
report_constraint(2)
```

```
reset_design(2)
```

remove_max_capacitance

Removes maximum capacitance limits from pins, ports, clocks, or designs.

SYNTAX

```
string remove_max_capacitance
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Provides a list of pins, ports, clocks, or designs from which to remove maximum capacitance limits.

DESCRIPTION

Removes maximum capacitance limits from pins, ports, clocks, or designs. A maximum capacitance limit is specified with the **set_max_capacitance** command.

The **report_constraint -max_capacitance** command shows maximum capacitance constraint evaluations. The **report_port -design_rule** command shows port maximum capacitance limits. The **report_design** command shows the default maximum capacitance setting for the current design.

EXAMPLES

The following example removes the maximum capacitance limit on ports "OUT*".

```
pt_shell> remove_max_capacitance [get_ports "OUT*"]
```

The following example removes the maximum capacitance limit on the current design.

```
pt_shell> remove_max_capacitance [current_design]
```

SEE ALSO

```
current_design(2)
remove_min_capacitance(2)
report_constraint(2)
report_design(2)
report_port(2)
get_ports(2)
set_max_capacitance(2)
remove_max_fanout(2)
remove_max_transition(2)
```

remove_max_fanout

Removes maximum fanout limits from ports or designs.

SYNTAX

```
status remove_max_fanout
  object_list
```

Data Types

```
object_list       list
```

ARGUMENTS

object_list

Lists the ports or designs from which to remove maximum fanout limits.

DESCRIPTION

Removes maximum fanout limits from ports or designs. A maximum fanout limit is specified with the **set_max_fanout** command.

The **report_constraint -max_fanout** command shows maximum fanout constraint evaluations. The **report_port -design_rule** command shows port maximum fanout limits. The **report_design** command shows the default maximum fanout setting for the current design.

EXAMPLES

The following example removes the maximum fanout limit on ports "OUT*".

```
pt_shell> remove_max_fanout [get_ports "OUT*"]
```

The following example removes the maximum fanout limit on the current design.

```
pt_shell> remove_max_fanout [current_design]
```

SEE ALSO

```
current_design(2)
get_ports(2)
report_constraint(2)
report_design(2)
report_port(2)
set_fanout_load(2)
set_max_capacitance(2)
set_max_fanout(2)
set_max_transition(2)
```

remove_max_time_borrow

Removes time borrow limit for latches.

SYNTAX

```
string remove_max_time_borrow
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Lists clocks, cells, data pins, or clock (enable) pins. If you specify a cell, all enable pins on that cell are affected.

DESCRIPTION

Removes maximum time borrow limit on objects. Time borrowing limits are specified with the **set_max_time_borrow** command. When the limit is removed, full time borrowing is allowed on level-sensitive latches.

To show time borrow limits, use the **report_exceptions** command.

EXAMPLES

The following example removes the maximum time borrow limit on clock PHI1.

```
pt_shell> remove_max_time_borrow [get_clocks PHI1]
```

The following example removes the maximum time borrow limit on all the pins of cells matching U1/latch*.

```
pt_shell> remove_max_time_borrow [get_cells U1/latch*]
```

SEE ALSO

```
report_exceptions(2)
set_max_time_borrow(2)
```

remove_max_transition

Removes maximum transition limits from pins, ports, clocks, or designs.

SYNTAX

```
status remove_max_transition
  object_list
```

Data Types

```
object_list        list
```

ARGUMENTS

object_list

Lists the pins, ports, clocks, or designs from which to remove maximum transition limits.

DESCRIPTION

This command removes maximum transition limits from pins, ports, clocks or designs. A maximum transition limit is specified with the **set_max_transition** command.

The **report_constraint -max_transition** command shows maximum transition constraint evaluations. The **report_port -design_rule** command shows port maximum transition limits. The **report_design** command shows the default maximum transition setting for the current design.

EXAMPLES

The following example removes the maximum transition limit on ports "OUT*".

```
pt_shell> remove_max_transition [get_ports "OUT*"]
```

The following example removes the maximum transition limit on the current design.

```
pt_shell> remove_max_transition [current_design]
```

SEE ALSO

```
current_design(2)
get_ports(2)
report_constraint(2)
report_design(2)
report_port(2)
set_max_capacitance(2)
set_max_fanout(2)
set_max_transition(2)
```

remove_min_capacitance

Removes minimum capacitance limits from ports or designs.

SYNTAX

```
string remove_min_capacitance
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Lists the ports or designs from which to remove minimum capacitance limits.

DESCRIPTION

Removes minimum capacitance limits from ports or designs. A minimum capacitance limit is specified with the **set_min_capacitance** command.

The **report_constraint -min_capacitance** command shows minimum capacitance constraint evaluations. The **report_port -design_rule** command shows port minimum capacitance limits. The **report_design** command shows the default minimum capacitance setting for the current design.

EXAMPLES

The following example removes the minimum capacitance limit on ports "OUT*".

```
pt_shell> remove_min_capacitance [get_ports "OUT*"]
```

The following example removes the minimum capacitance limit on the current design.

```
pt_shell> remove_min_capacitance [current_design]
```

SEE ALSO

```
current_design(2)
remove_max_capacitance(2)
report_constraint(2)
report_design(2)
report_port(2)
get_ports(2)
set_min_capacitance(2)
set_max_capacitance(2)
set_max_fanout(2)
set_max_transition(2)
```

remove_min_pulse_width

Removes a previously-specified minimum pulse width constraint from specified design objects.

SYNTAX

```
string remove_min_pulse_width
   [-low]
   [-high]
   [object_list]
```

Data Types

```
object_list      list
```

ARGUMENTS

-low

Indicates that the minimum pulse width constraint is to be removed only for low clock signal levels. If you do not specify the *-low* or *-high* option, the constraint is removed for both low and high pulses of clock signals.

-high

Indicates that the minimum pulse width constraint is to be removed only for high clock signal levels. If you do not specify the *-low* or *-high* option, the constraint is removed for both low and high pulses of clock signals.

object_list

Specifies a list of clocks, cells, pins, or ports in the current design for which the minimum pulse width constraint is to be removed. If you specify a cell, all pins on that cell are affected. If you do not specify any objects, the minimum pulse width check is removed from all objects in the current design.

DESCRIPTION

The **remove_min_pulse_width** command removes the pulse width check previously specified by the **set_min_pulse_width** command for clock signals in clock trees or at sequential devices.

To generate a report of pulse width constraints, use the **report_constraint -min_pulse_width** or **report_min_pulse_width** command.

The **reset_design** command removes all user-specified attributes from a design, including those set by the **set_min_pulse_width** command.

EXAMPLES

The following example removes a minimum pulse width requirement previously set for clock CK1.

```
pt_shell> remove_min_pulse_width [get_clocks CK1]
```

The following example removes a minimum pulse width requirement previously set for pin U1/Z.

```
pt_shell> remove_min_pulse_width U1/Z
```

SEE ALSO

current_design(2)
report_constraint(2)
report_min_pulse_width(2)
reset_design(2)
set_min_pulse_width(2)

remove_mtcmos_switch_cell_sequence

Removes previously created mtcmos switch cell event sequence.

SYNTAX

```
status remove_mtcmos_switch_cell_sequence
   [-noexpand]
   supply_nets
```

Data Types

```
list supply_nets
```

ARGUMENTS

-noexpand

Avoids expanding the listed *supply_nets* to include the virtual supply nets that are connected by MTCMOS switch cells. The MTCMOS cells are also referred as power management (PM) cells. By default, the command expands the list of supply nets.

supply_nets

The event sequence of mtcmos switch cells, which are connected to the supply nets, is removed. The list of supply nets is automatically expanded to include the virtual supply nets that are connected to those listed by MTCMOS switch cells. To avoid the expansion, use the -noexpand option.

DESCRIPTION

The **remove_mtcmos_switch_cell_sequence** command removes mtcmos switch cell event sequence that are previously created using the **create_mtcmos_switch_cell_sequence** command.

EXAMPLES

The following example shows how to remove event sequence of all mtcmos switch cell connected to supply net VDD:

```
pr_shell> remove_mtcmos_switch_cell_sequence VDD
```

SEE ALSO

```
create_mtcmos_switch_cell_sequence(2)
report_mtcmos_switch_cell_sequence(2)
```

remove_mw_cel

Removes Milkyway designs from the design library.

SYNTAX

```
status remove_mw_cel
   [-hierarchy [-check_only] ]
   [-version_kept count]
   [-verbose]
   [-all_versions]
   [-all_view]
   mw_cel_list
```

Data Types

```
count          integer
mw_cel_list    list
```

ARGUMENTS

-hierarchy

Deletes the specified top-level design and all of its subdesigns. By default, the command deletes only the specified designs and retains any subdesigns.

When you use the -hierarchy option, the *mw_cel_list* argument must specify only a single Milkyway design; the specified view name and version number are ignored. Using the -hierarchy option removes all views and lower-level designs in CEL views and FILL views; the option also removes all versions you choose to keep the latest version by using the -version_kept 1 option.

The -hierarchy option cannot be used with the -all_views or -all_versions option.

For hierarchical FILL views, the command operates on the whole hierarchy by default, regardless of the -hierarchy option setting.

-check_only

This option is valid only when used with the -hierarchy option. The option lists the Milkyway designs (including versions and views) to be removed without actually removing any data. Use this option when you want to check what will be removed by the -hierarchy option.

-version_kept *count*

Specifies the versions to be kept. The value must be an integer greater than or equal to *0*. The value *0* removes all versions; the value *1* keeps the most recent version and removes the rest.

When used with the -hierarchy option, you can set the -version_kept option to *1* (to keep the most recent

version only) or *0* (to delete all versions). No other value is allowed.

-verbose

Reports the possible cause when the command fails to remove the specified Milkyway designs. This option is valid only when used with the -hierarchy option.

-all_versions

Removes all versions of the specified Milkyway designs. This option is ignored when the -hierarchy option is used.

-all_view

Removes all views of the specified Milkyway designs, including CEL views, FRAM views, ERR views, and so on. By default, the command removes only the specified type of view. This option is ignored when the -hierarchy option is used.

mw_cel_list

Specifies the Milkyway designs to be removed. You can specify Milkyway design views by name, pattern or collection, as shown in the EXAMPLES section below.

When used with the -hierarchy option, the specified design must be a single top-level design. The command operates on CEL views as well as all other supported views, so any specified view name or version number is ignored.

When used without the -hierarchy option, only the specified view and version are deleted. Older versions are retained unless the -version_kept option is specified.

DESCRIPTION

This command removes the specified Milkyway designs from the current library. Before removing the designs, you must make sure the designs are closed.

The -version_kept option allows you to remove older versions of the Milkyway design while keeping newer ones.

The -hierarchy option allows you to remove or purge a Milkyway design and its whole hierarchical tree. Note that using this option changes the default behavior of the command with respect to version numbers. Without the -hierarchy option, the command deletes only the specified version number or the most recent version (if no version number is specified) and retains all older versions. With the -hierarchy option, the command deletes all versions unless you retain the most recent version by using the version_kept 1 option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the latest version of the Milkyway designs whose names start with *IP*.

```
pr_shell> remove_mw_cel IP* -verbose
Information: Removed design IP_test_TOP.CEL.
```

```
Information: Removed design IP_test1.CEL.
1
```

The following example removes version **1** of the design named *IP_test*. In this case, *IP_test* has two versions. After removing version 1, you can get a Milkyway design collection based on its remaining version.

```
pr_shell> remove_mw_cel "IP_test.CEL;1" -verbose
Information: Removed design IP_test.CEL.
1
pr_shell> list_mw_cels
IP_test_TOP
IP_test
```

The following example removes the design named *IP_test_TOP* and all of its subdesigns from the current design library.

```
pr_shell> remove_mw_cel -hierarchy IP_test_TOP
Starting hierarchical delete ...
Deleting the cell: IP_test ...
Deleting the cell: IP_test_TOP ...
1
```

The following example purges old versions of the design named *IP_test_TOP* and all of its subdesigns from the current design library. Only the latest version of these designs is kept.

```
pr_shell> remove_mw_cel -hierarchy -version_kept 1 \
IP_test_TOP
Starting hierarchical purge ...
Purging the cell: IP_test ...
Purging the cell: IP_test_TOP ...
1
```

SEE ALSO

```
close_mw_cel(2)
copy_mw_cel(2)
create_mw_cel(2)
current_mw_cel(2)
open_mw_cel(2)
save_mw_cel(2)
```

remove_net

Removes nets from the current design.

SYNTAX

```
int remove_net
   -all
   net_list
```

Data Types

```
net_list      list
```

ARGUMENTS

-all

Indicates that all nets are to be removed from the current design. The **-all** and *net_list* options are mutually exclusive; you can specify only one.

net_list

Specifies a list of nets to be removed from the current design. The **-all** and *net_list* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **remove_net** command removes specified nets or all nets from the current design. Like all other netlist editing commands, for the **remove_net** command to succeed, all of its arguments must succeed. If any argument fails, the netlist remains unchanged. The **remove_net** command returns a 1 if successful and 0 if unsuccessful.

Each net specified must be in scope; that is, at or below the current instance.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design,

rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following example, nets are removed if their names begin with "new":

```
pt_shell> remove_net [get_nets new*]
Information: Removed net 'new_net1'. (NED-017)
Information: Removed net 'new_net2'. (NED-017)
1
```

SEE ALSO

size_cell(2)
swap_cell(2)
write_changes(2)

remove_noise_immunity_curve

Removes noise immunity curve for a library pin or port.

SYNTAX

```
int remove_noise_immunity_curve
   [-above]
   [-below]
   [-low]
   [-high]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-above

Removes the noise immunity curve for above ground or power rail noise analysis region.

-below

Removes the noise immunity curve for below ground or power rail noise analysis region.

-low

Removes the noise immunity curve for ground rail noise.

-high

Removes the noise immunity curve for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

DESCRIPTION

This command removes the noise immunity curve information set by the **set_noise_immunity_curve** command.

EXAMPLES

This example removes noise immunity curve information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_noise_immunity_curve -above lsi_10k/IV/A
```

SEE ALSO

set_noise_immunity_curve(2)

remove_noise_lib_pin

Removes an equivalent noise library pin for a driver or load.

SYNTAX

```
int remove_noise_lib_pin
   pins
```

Data Types

```
pins      list
```

ARGUMENTS

pins

Specifies the collection of pins for which the equivalent noise library pin is set by the **set_noise_lib_pin** command that needs to be removed.

DESCRIPTION

Given a collection of pins, this command allows you to remove a noise library pin previously set as an equivalent in terms of library noise information.

EXAMPLES

The following example removes any equivalent noise information on two input pins of the design, which is the same as an input library pin:

```
pt_shell> remove_noise_lib_pin [get_pins {cell/pin1 cell/pin2}] \
```

SEE ALSO

```
set_noise_lib_pin(2)
update_noise(2)
```

```
report_noise(2)
report_noise_calculation(2)
```

remove_noise_margin

Removes noise margin for a library pin or port.

SYNTAX

```
int remove_noise_margin
   [-above]
   [-below]
   [-low]
   [-high]
   object_list
```

Data Types

```
object_list      list
```

ARGUMENTS

-above

Removes the noise margin for above ground or power rail noise analysis region.

-below

Removes the noise margin for below ground or power rail noise analysis region.

-low

Removes the noise margin for ground rail noise.

-high

Removes the noise margin for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

DESCRIPTION

This command removes the noise margin information set by the **set_noise_margin** command.

EXAMPLES

This example removes noise margin information for above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_noise_margin -above lsi_10k/IV/A
```

SEE ALSO

set_noise_margin(2)

remove_operating_conditions

Removes operating conditions from current design, cells, or ports.

SYNTAX

```
string remove_operating_conditions
   [-object_list objects]
```

Data Types

```
objects      list
```

ARGUMENTS

-object_list objects

Specifies the cells or ports to remove operating conditions from. The command undoes operating conditions set by the **set_operating_conditions** command. Without the *-object_list* option, all the operating conditions are removed from the design. Both leaf cells and hierarchical blocks are accepted with the *-object-list* option.

DESCRIPTION

Removes operating conditions settings from the current design, individual blocks, leaf cells, or ports.

EXAMPLES

This example removes all operating conditions from the current design. Both design-specific and cell-specific operating conditions are removed.

```
pt_shell> remove_operating_conditions
```

SEE ALSO

set_operating_conditions(2)

remove_output_delay

Removes output delay from output ports or pins.

SYNTAX

```
string remove_output_delay
   [-clock clock_name]
   [-clock_fall]
   [-level_sensitive]
   [-rise]
   [-fall]
   [-max]
   [-min]
   port_pin_list
```

Data Types

clock_name	string
port_pin_list	list

ARGUMENTS

-clock *clock_name*

Relative clock; {""} for input delay relative to no clock.

-clock_fall

Removes the delay relative to falling edge of clock. If you specify the *clock_name* variable without the **-clock_fall** command, the delay relative to rising edge of the clock is removed.

-level_sensitive

Removes level-sensitive output delay.

-rise

Removes rising output delay.

-fall

Removes falling output delay.

-max

Removes maximum output delay.

-min

Removes minimum output delay.

port_pin_list

Specifies a list of ports and pins. Each element in the list is either a collection of ports or pins, or a pattern which matches ports or pins on the current design.

DESCRIPTION

Removes output delay values for objects in the current design. Set output delay with the **set_output_delay** command. By default, all output delays on each object in the **port_pin_list** command are removed. To restrict the removed output delay values, use the *-clock*, *-clock_fall*, *-min*, *-max*, *-rise*, or *-fall* options. To show output delays associated with ports, use the **report_port -output_delay** command.

EXAMPLES

The following example removes all output delay values from all output ports in the current design.

```
pt_shell> remove_output_delay [all_outputs]
```

The following example removes output maximum rise delay values from OUT1, OUT3, and BIDIR12.

```
pt_shell> remove_output_delay -max -rise { OUT1 OUT3 BIDIR12 }
```

The following example removes all output delays for port OUT1 relative to the falling edge of CLK2.

```
pt_shell> remove_output_delay -clock CLK2 -clock_fall OUT1
```

The following example removes all output delays not relative to any clock for port OUT2.

```
pt_shell> remove_output_delay -clock {""} OUT2
```

SEE ALSO

```
all_outputs(2)
collections(2)
report_port(2)
set_output_delay(2)
set_input_delay(2)
```

remove_parasitic_corner

Removes a previously set parasitic corner in the presence of variation-aware parasitics.

SYNTAX

```
status remove_parasitic_corner
```

ARGUMENTS

None.

DESCRIPTION

The **remove_parasitic_corner** command removes the parasitic corner information previously set via the **set_parasitic_corner** command. The command succeeds only if variation-aware parasitics were annotated and the **set_parasitic_corner** command is used to set the parasitic corner.

After using this command, all the parasitic attributes and analysis use nominal values instead of corner values.

EXAMPLES

The following example removes the parasitics corner information set previously.

```
pt_shell> remove_parasitic_corner
```

SEE ALSO

```
set_parasitic_corner(2)
read_parasitics(2)
report_annotated_parasitics(2)
```

remove_path_group

Removes path group objects.

SYNTAX

```
string remove_path_group
   -all
   path_group_list
```

Data Types

```
path_group_list        list
```

ARGUMENTS

-all

Removes all path groups in the current design.

path_group_list

Specifies path group names to remove. Each element in the list is either a collection of path groups or a pattern matching path group names.

DESCRIPTION

Removes path groups in the current design. The **group_path** and **create_clock** commands create path groups. Path groups affect the cost function for optimization and influence the timing reports. If you remove a path group, any paths in that group are implicitly assigned to the default path group. To show path group information, use the **report_path_group** command.

EXAMPLES

The following example removes all path groups in the design.

```
pt_shell> remove_path_group -all
```

The following example removes path group "BUS1".

```
pt_shell> remove_path_group BUS1
```

SEE ALSO

collections(2)
create_clock(2)
group_path(2)
report_path_group(2)
report_timing(2)

remove_port_fanout_number

Removes fanout number information on ports.

SYNTAX

```
string remove_port_fanout_number
   port_list
```

Data Types

port_list	list

ARGUMENTS

port_list

Specifies a list of ports. Each element in the list is either a collection of ports or a pattern that matches ports on the current design.

DESCRIPTION

Removes the **fanout_number** settings on ports. The **set_port_fanout_number** command sets the number of external fanout pins on ports, which is used along with wire load models to calculate capacitance and resistance of nets. To show port fanout number information, use the **report_port -wire_load** command.

EXAMPLES

This example removes the port fanout_number settings from ports OUT1*.

```
pt_shell> remove_port_fanout_number [get_ports OUT1*]
```

SEE ALSO

collections(2)
get_ports(2)

```
report_port(2)
set_port_fanout_number(2)
set_wire_load_model(2)
```

remove_power_groups

Remove the existing power groups.

SYNTAX

```
string remove_power_groups
   -all
   group_names
```

Data Types

```
group_names         list
```

ARGUMENTS

-all

Removes all the user-defined power groups. Note that predefined power groups are note deleted using this option.

group_names

Specifies the groups to be removed.

DESCRIPTION

The **remove_power_groups** command removes some or all power groups that are previously created but no longer interested. The predefined power groups can be removed, but not with the *-all* option. The predefined power group names need to be explicitly specified.

EXAMPLES

In the following example, all the user defined power groups are removed.

```
pt_shell> remove_power_groups -all
1
```

In the following example, the predefined clock_network power group is removed.

```
pt_shell> remove_power_groups clock_network
1
```

SEE ALSO

create_power_group(2)
report_power_groups(2)
get_power_group_objects(2)

remove_power_management_mtcmos_locations

Removes the power management location (PML) file from the specified cell.

SYNTAX

```
status remove_power_management_mtcmos_locations
   -cell cell_name
   -library library_name
```

Data Types

```
string cel_name
string library_name
string net_name
```

ARGUMENTS

-cell *cel_name*

Specifies the name of the Milkyway CEL design whose power management location information is to be removed.

-library *library_name*

Specifies the Milkyway design library in which the Milkyway CEL view exists.

DESCRIPTION

This command removes the PML file from the specified cell. The PML file describes the on-resistance source and drain locations of power management cells.

EXAMPLES

The following example removes the PML file from the PMCell cell.

```
pr_shell> remove_power_management_mtcmos_locations -library MyLib -cell PMCell
1
```

SEE ALSO

import_power_management_mtcmos_locations(2)
export_power_management_mtcmos_locations(2)

remove_probes

Remove voltage and current probes from pg_pins.

SYNTAX

```
string remove_probes
   [pg_pins]
```

Data Types

```
list pg_pins
```

ARGUMENTS

pg_pins

Specifies a list of pg_pins from which voltage and current probes are to be removed. If not specified, all pg_pins in the design are inspected and their probes are removed if installed.

DESCRIPTION

The **remove_probes** command removes voltage and current probes installed at pg_pins. If *pg_pins* argument is specified, probes at the specified pg_pins, if installed, are removed. Otherwise, all pg_pins in the design are inspected and probes are removed from the pg_pins if installed.

EXAMPLES

The following example shows how to remove voltage and current probes from the specified pg_pins.

```
pr_shell> remove_probes [get_pg_pins INV*/vdd]
Information: Voltage and current probes removed from 12 pg_pins.
1
```

SEE ALSO

```
set_probes(2)
report_probes(2)
```

remove_propagated_clock

Removes a propagated clock specification.

SYNTAX

```
string remove_propagated_clock
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Lists clocks, ports, or pins.

DESCRIPTION

Removes propagated clock specification from clocks, ports, or pins in the current design. The **set_propagated_clock** command specifies that delays be propagated through the clock network to determine latency at register clock pins. If this is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a user specified latency (set by the **set_clock_latency** command) or zero latency, by default. Propagated clock latency is normally used for post layout, after final clock tree generation.

Ideal clock latency provides an estimate of the clock tree for prelayout. You can also use the **set_clock_latency** command to specify an ideal latency, which overrides the propagated clock specification for an object.

For information about propagated clock attributes on clocks, see the **report_clock** man page.

EXAMPLES

The following example removes propagated clock specifications from all clocks in the design.

```
pt_shell> remove_propagated_clock [all_clocks]
```

SEE ALSO

```
report_clock(2)
set_clock_latency(2)
set_propagated_clock(2)
```

remove_pulse_clock_max_transition

Removes maximum transition limits from pulse clock network and input of pulse generator.

SYNTAX

```
string remove_pulse_clock_max_transition
   [-rise]
   [-fall]
   [-transitive_fanout]
   object_list
```

Data Types

```
object_list       list
```

ARGUMENTS

-rise

Removes the rise transition constraint.

-fall

Removes falling transition constraint.

-transitive_fanout

Removes the constraint from the transitive fanout of the pulse generator. If this option is not specified, the constraint is removed from the input of pulse generator.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock, and design from which to remove maximum pulse clock transition limits.

DESCRIPTION

Removes maximum pulse clock transition limits from the specified objects. A maximum transition limit is specified with the **set_pulse_clock_max_transition** command. This command can change the constraint on the object, which has conflicting constraints due to overlap. You can use the *-rise* and *-fall* options to remove only the rising or the falling transition respectively. If neither the *-rise* nor the *-fall* option is specified, both rise and fall transition limits are removed. You must specify the *-transitive_fanout* option to remove pulse clock maximum transition constraint from the transitive fanout of pulse generators.

EXAMPLES

The following example removes the pulse clock maximum transition limit from the input of all the cells corresponding to the library cell pulse_rise_high in the library celllib.

```
pt_shell> remove_pulse_clock_max_transition {"celllib/pulse_rise_high"}
```

The following example removes maximum transition limit from all the pulse clock networks in the clock network clk

```
pt_shell> remove_pulse_clock_max_transition -transitive_fanout [get_clocks clk]
```

SEE ALSO

```
current_design(2)
set_pulse_clock_max_transition(2)
report_pulse_clock_max_transition(2)
report_constraint(2)
```

remove_pulse_clock_max_width

Removes maximum pulse width limits from pulse clock network.

SYNTAX

```
string remove_pulse_clock_max_width
   [-transitive_fanout]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-transitive_fanout

Remove the constraints from the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove maximum pulse clock width limits.

DESCRIPTION

Removes maximum pulse clock width limits from the specified objects. A maximum width limit is specified with the **set_pulse_clock_max_width** command. This command can change the constraint on the object, which has conflicting constraints due to overlap.

EXAMPLES

The following example removes maximum width limit from all the pulse clock networks in the design.

```
pt_shell> remove_pulse_clock_max_width [current_design]
```

SEE ALSO

```
current_design(2)
set_pulse_clock_max_width(2)
report_pulse_clock_max_width(2)
report_constraint(2)
```

remove_pulse_clock_min_transition

Removes minimum transition limits from input of pulse generator.

SYNTAX

```
string remove_pulse_clock_min_transition
   [-rise]
   [-fall]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-rise

Removes the rise transition constraint.

-fall

Removes the falling transition constraint.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove minimum pulse clock transition limits.

DESCRIPTION

Removes minimum pulse clock transition limits from the input of specified objects. If clock or design is specified, the minimum transition limit is removed from the input of all the pulse generators in the clock network or design, respectively. A minimum transition limit is specified with the **set_pulse_clock_min_transition** command. This command can change the constraint on the object, which has conflicting constraints due to overlap. You can use the *-rise* and *-fall* options to remove only the rising or the falling transition, respectively. If neither the *-rise* or *-fall* option is specified, both rise and fall transition limits are removed.

EXAMPLES

The following example removes a minimum transition limit from input of all the cells corresponding to the *pulse_rise_high* library cell in the *celllib* library.

```
pt_shell> remove_pulse_clock_min_transition {"celllib/pulse_rise_high"}
```

SEE ALSO

```
current_design(2)
set_pulse_clock_min_transition(2)
report_pulse_clock_min_transition(2)
report_constraint(2)
```

remove_pulse_clock_min_width

Removes minimum pulse width limits from pulse clock network.

SYNTAX

```
string remove_pulse_clock_min_width
   [-transitive_fanout]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-transitive_fanout

Removes the constraints from the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove minimum pulse clock width limits.

DESCRIPTION

Removes minimum pulse clock width limits from the specified objects. A minimum width limit is specified with the **set_pulse_clock_min_width** command. This command can change the constraint on the object, which has conflicting constraints due to overlap.

EXAMPLES

The following example removes minimum width limit from all the pulse clock networks in the clock network clk.

```
pt_shell> remove_pulse_clock_min_width [get_clocks clk]
```

SEE ALSO

```
current_design(2)
set_pulse_clock_min_width(2)
report_pulse_clock_min_width(2)
report_constraint(2)
```

remove_qtm_attribute

Removes an attribute setting from the QTM object.

SYNTAX

```
string remove_qtm_attribute
   -class lib | lib_cell | lib_pin
   attribute_name
   object_names
```

Data Types

```
attribute_name     string
object_names       list
```

ARGUMENTS

-class lib | lib_cell | lib_pin

Specifies the class of the QTM objects to remove attribute from.

attribute_name

Specifies the name of the attribute.

object_names

Specifies the names of the objects from which to remove the named attribute. Each element in the list must be a name to identify an object in the specified class. The object names should not be specified when the object class is either **lib** or **lib_cell**. It must be specified when the class is **lib_pin**, in which case the object names should be the names of ports already defined for the QTM.

DESCRIPTION

The **remove_qtm_attribute** command removes attributes you set with the **set_qtm_attribute** command.

You can remove either application attributes or user attributes you defined for QTM object with the **define_qtm_attribute** command.

EXAMPLES

The following example defines the is_XYZ Boolean attribute for QTM cell, then sets the value on the QTM cell under construction. Finally, it removes the attribute from the cell.

```
pt_shell> define_qtm_attribute -type boolean -class lib_cell "is_XYZ"
pt_shell> set_qtm_attribute -class lib_cell "is_XYZ" "true"
pt_shell> remove_qtm_attribute -class lib_cell "is_XYZ"
```

The following example defines the my_float_attr user attribute for all QTM port objects, sets the value for two QTM ports, and removes the attribute from one of the ports.

```
pt_shell> define_qtm_attribute -type float -class lib_pin "my_float_attr"
pt_shell> set_qtm_attribute -class lib_pin "my_float_attr" 100.0 {IN, OUT}
pt_shell> remove_qtm_attribute -class lib_pin "my_float_attr" {OUT}
```

SEE ALSO

define_qtm_attribute(2)
define_user_attribute(2)
set_qtm_attribute(2)

remove_rail_current

Removes PG pin electrical current assigned by the **set_rail_current** command.

SYNTAX

```
status remove_rail_current
   object_list
   [-static]
   [-waveform]
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

A list of PG pins to which to remove electrical current. Each item in the list can be a PG name pattern or collection.

-static

The static electrical current to be removed.

-waveform

The dynamic current waveform to be removed.

DESCRIPTION

The **remove_rail_current** command removes PG pin electrical current previously assigned by the **set_rail_current** command. If both options *-static* and *-waveform* are not specified, this command will remove both static current and current waveform previously assigned. This command returns 1 if successful, and 0 if unsuccessful.

If the effect of the **set_rail_current -static** command was to override current and its corresponding power values calculated by the **update_currents** command, then those calculated current and power values are restored when **remove_rail_current** is executed.

After this command completes, the PG pin *static_current or current waveform* attribute for the affected PG pins will reflect the calculated static current and current waveform for the PG pin if **update_currents** has been executed.

EXAMPLES

The following example assigns an electrical current to the PG pin VDD of cell instance MyCell, and then removes the assigned static current to all the PG pins:

```
pr_shell> set_rail_current -static -1.0 MyCell/VDD

pr_shell> remove_rail_current [get_pg_pins MyCell/*]
```

SEE ALSO

```
set_rail_current(2)
set_rail_power(2)
report_rail_power(2)
update_currents(2)
```

remove_rail_macro_mode_sequence

Removes the macro mode sequence from the PG pins of the hard macros.

SYNTAX

```
status remove_rail_macro_mode_sequence
   object_list
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

Specifies the hard macro PG pins whose user-assigned mode sequence is to be removed.

DESCRIPTION

This command removes the mode sequence from the PG pins of the hard macros. The mode sequence is set by using the **set_rail_macro_mode_sequence** command. The command returns 1 if successful, and 0 if unsuccessful.

EXAMPLES

The following example removes a user-assigned mode sequence from the PG pin inst4/VDD:

```
pr_shell> remove_rail_macro_mode_sequence inst4/VDD
1
```

SEE ALSO

```
rail_attributes(2)
set_rail_macro_mode_sequence(2)
```

```
update_currents(2)
```

remove_rail_maps

Frees all memory allocated for the display of rail maps in the GUI.

SYNTAX

status **remove_rail_maps**

DESCRIPTION

When showing result data (voltage drop maps, resistivity maps, power maps, etc.) the GUI allocates memory to achieve better performance when visualizing the data. This memory can be large and is no longer needed when you are done looking at the maps. You can use this command to make that memory available to the session again.

If this command is run while a rail map is being displayed in the GUI then the map is cleared and you are returned to the layout view of the design.

EXAMPLES

In the following example a rail map is viewed in the GUI, and then the memory used for the map display is recovered.

```
pr_shell> start_gui
   <...Display a rail map...>
pr_shell> remove_rail_maps
```

SEE ALSO

start_gui(2)

remove_rail_power

Removes PG pin power assigned by the **set_rail_power** command.

SYNTAX

```
status remove_rail_power
   [-supply_net supply_net]
   object_list
```

Data Types

```
list object_list
list supply_net
```

ARGUMENTS

object_list

A list of cells, lib_cells or pg_pins on which to remove rail power.

-supply_net *supply_net*

Limits the command to act only on items in the *object_list* that are connected to the specified *supply_net(s).* If no *supply_net* is specified for a given cell or lib_cell, then the power value is removed from all PG pins of the object to which power has been applied by **set_rail_power**. This option is valid only for *object_list* items that are cells or lib_cells.

DESCRIPTION

The **remove_rail_power** command removes PG pin power previously assigned by **set_rail_power**. The PG pins can be directly specified, or can be indirectly specified by referencing their associated cells, lib_cells or attached supply nets. The command returns 1 if successful, and 0 if unsuccessful.

If the effect of the **set_rail_power** command was to override power values calculated during **update_currents**, then those calculated power values are restored when **remove_rail_power** is executed.

If a PG pin is not specified with this command, either directly or by specifying the attached supply net, then the assigned power is removed from all the PG pins of the specified cells or lib_cells. When power is removed from lib_cell PG pins, all instances of the lib_cell will have the power removed.

After this command completes, the PG pin *total_power* attribute for the affected PG pins will reflect the calculated total power for the PG pin if **update_currents** has been executed. The user can also use the **report_rail_power** command to verify that the assigned power is removed, where the PG pin whose power has

been removed does not show up.

EXAMPLES

The following example sets the total power for the vdd pin of a specific blackbox instance, and then removes the applied power to all pins.

```
pr_shell> set_rail_power .01 U153/vdd
pr_shell> remove_rail_power U153
```

SEE ALSO

rail_attributes(2)
report_rail_power(2)
set_rail_power(2)
update_currents(2)

remove_rail_result

Removes a rail result data set from the current session.

SYNTAX

```
status remove_rail_result
   result_name
```

Data Types

```
string result_name
```

ARGUMENTS

result_name

Specifies the name of the rail result to be removed.

DESCRIPTION

The **remove_rail_result** command removes the named rail result from the current session. Rail results are generated by certain rail analysis commands, such as **calculate_rail_voltage** and **calculate_minimum_path_resistances**, and are cached to the disk only during the session. If you execute multiple analysis commands during the same session, you might have results that are no longer needed.

To remove the rail results that are no longer needed during the session, use the **remove_rail_result** command. To list all the rail results in the current session, run the **report_rail_result** command.

Rail results are not persistent between sessions, meaning that if you exit the tool without saving the rail results and then invoke the tool again, your rail results will be gone.

If you want to clear rail results at a point in the current session rather than delete them altogether, use the **current_rail_result -clear** command instead. The results can be accessed at a later point in the session.

EXAMPLES

This example first clears a rail result named VDROP_RAIL_RESULT_1 from the memory disk, then removes it entirely from the memory and cached disk for the current session.

```
pr_shell> current_rail_result
VDROP_RAIL_RESULT_1


pr_shell> report_rail_result -verbose
****************************************
Report : rail analysis results
Design : MyDesign
Version: E-2010.12
Date   : Wed Aug 11 17:52:41 2010
****************************************

   VDROP_RAIL_RESULT_1
      Nets in this result:
      VDD
      VSS


   VDROP_RAIL_RESULT_2
      Nets in this result:
      VDD
      VSS

1
pr_shell> current_rail_result -clear
none
pr_shell> report_rail_result -verbose
****************************************
Report : rail analysis results
Design : MyDesign
Version: E-2010.12
Date   : Wed Aug 11 17:52:41 2010
****************************************

   VDROP_RAIL_RESULT_1
      Nets in this result:
      VDD
      VSS


   VDROP_RAIL_RESULT_2
      Nets in this result:
      VDD
      VSS

1
pr_shell> remove_rail_result VDROP_RAIL_RESULT_1
1
pr_shell> current_rail_result
none
pr_shell> report_rail_result -verbose
****************************************
Report : rail analysis results
Design : MyDesign
Version: E-2010.12
Date   : Wed Aug 11 17:52:41 2010
****************************************

   VDROP_RAIL_RESULT_2
      Nets in this result:
      VDD
      VSS

1
pr_shell> current_rail_result  VDROP_RAIL_RESULT_2
```

```
VDROP_RAIL_RESULT_2
```

SEE ALSO

```
calculate_rail_voltage(2)
calculate_minimum_path_resistances(2)
current_rail_result(2)
rename_rail_result(2)
report_rail_results(2)
```

remove_rail_voltage

Removes power rail voltage that was set by the **set_rail_voltage** command on cells.

SYNTAX

```
int remove_rail_voltage
   cell_list
```

Data Types

```
cell_list          list
```

ARGUMENTS

cell_list

Specifies a list of cells from which to remove rail voltages.

DESCRIPTION

Removes power rail voltage that was set by the **set_rail_voltage** command on cells. If dynamic components of rail voltage have been specified they are also removed along with the total rail voltage. This command "undoes" the voltages specified by the **set_rail_voltage** command. It can be applied to both leaf cells and hierarchical blocks.

EXAMPLES

The following example removes rail voltages from a cell named *h1/u3*.

```
pt_shell> remove_rail_voltage [get_cells {h1/u3}]
1
```

SEE ALSO

```
set_operating_conditions(2)
```

```
set_rail_voltage(2)
```

remove_resistance

Removes resistance on nets.

SYNTAX

```
status remove_resistance
   net_list
```

Data Types

```
net_list     list
```

ARGUMENTS

net_list

Specifies a list of nets in the current design, whose resistances are removed.

DESCRIPTION

Specifies that the lumped resistance annotated on the list of nets must be removed. PrimeTime will revert to using the resistance from detailed parasitics (set using the **read_parasitics** command), if they exist. If not, the internally estimated net resistance is used.

Annotated resistances can also be removed on the full design by using the **reset_design** command.

EXAMPLES

The following example removes the annotated resistance on net w23.

```
pt_shell> remove_resistance [get_nets "w23"]
```

SEE ALSO

all_outputs(2)

```
current_design(2)
set_resistance(2)
report_net(2)
report_port(2)
reset_design(2)
set_drive(2)
```

remove_sense

Removes sense information defined on pins or cell timing arcs.

SYNTAX

```
status remove_sense
   [-type clock | data]
   [-clocks clock_list]
   [-all]
   object_list
```

Data Types

```
clock_list      list
object_list     list
```

ARGUMENTS

-type clock | data

Specifies the removal of sense from **clock** (the default) or **data** networks.

-clocks *clock_list*

Specifies a list of clock objects to be associated with the given pin objects in the *object_list*. This option removes the unateness only for the specified clock domain. This option can only remove clock sense predefined by the **set_sense -type clock** command. It does not remove the default clock sense setting for this given pin.

-all

Removes all unateness information in current design.

object_list

Specifies a list of pins or cell timing arcs from which to remove predefined unateness.

DESCRIPTION

This command removes the unateness information that was previously defined by the **set_sense** command.

EXAMPLES

This example removes positive unateness defined on the XOR/Z pin with respect to the CLK1 clock but does not remove the negative unateness.

```
pt_shell> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
pt_shell> set_sense -negative XOR/Z
pt_shell> remove_sense -clocks [get_clocks CLK1] XOR/Z
```

This example removes all unateness information in the current design.

```
pt_shell> remove_sense -all
```

SEE ALSO

set_sense(2)

remove_setup_hold_pessimism_reduction

Removes the optimization constraints for setup-hold pessimism reduction.

SYNTAX

```
status remove_setup_hold_pessimism_reduction
   [-setup_cutoff]
   [-hold_cutoff]
```

ARGUMENTS

-setup_cutoff

Resets the setup cutoff slack value to negative INFINITY.

-hold_cutoff

Resets the hold cutoff slack value to negative INFINITY.

DESCRIPTION

This command removes the optimization constraints for setup-hold pessimism reduction (SHPR). Using this command without options disables SHPR optimization.

EXAMPLES

The following example resets the setup cutoff to negative INFINITY:

```
pt_shell> remove_setup_hold_pessimism_reduction -setup
1
```

The following example resets the hold_cutoff to negative INFINITY:

```
pt_shell> remove_setup_hold_pessimism_reduction -hold
1
```

The following example resets both hold_cutoff and setup_cutoff to negative INFINITY:

```
pt_shell> remove_setup_hold_pessimism_reduction -hold -setup
1
```

The following example disable SHPR optimization:

```
pt_shell> remove_setup_hold_pessimism_reduction
1
```

SEE ALSO

set_setup_hold_pessimism_reduction(2)

remove_si_aggressor_exclusion

Removes the exclusive groups set by the **set_si_aggressor_exclusion** command.

SYNTAX

```
status remove_si_aggressor_exclusion
   [-rise]
   [-fall]
   [-all]
   [nets]
```

Data Types

```
nets        list
```

ARGUMENTS

-rise

Removes the exclusive group set for the aggressor nets *anets* in the *rise* direction. If neither the **-rise** nor the **-fall** option is specified, the exclusive groups of the aggressor nets *anets* in both **-rise** and **-fall** directions are removed.

-fall

Removes the exclusive group set for the aggressor nets *anets* in the *fall* direction. If neither the **-rise** nor the **-fall** option is specified, the exclusive groups of the aggressor nets *anets* in both **-rise** and **-fall** directions are removed.

-all

Removes all the exclusive groups set on the design.

nets

Specifies the list of nets belonging to one exclusive group that are to be removed.

DESCRIPTION

The **remove_si_aggressor_exclusion** command removes the effect of the **set_si_aggressor_exclusion** command that was set on the aggressor nets. Only those groups that have been set can be removed. A warning message is issued if you are trying to remove a group that has not been set.

These exclusive commands are independent of parasitics, so they can be applied even before reading in parasitics.

Note that application of exclusive aggressors are not done incrementally. The next **update_timing** and **update_noise** commands would produce a full update.

You can use the **report_si_aggressor_exclusion** command to check whether the **remove_si_aggressor_exclusion** command was applied to the exclusive groups.

EXAMPLES

The following example shows how to remove the exclusion on nets *SCAN_LOGIC** for rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
pt_shell> remove_si_aggressor_exclusion [get_nets SCAN_LOGIC*]
1
```

The following example shows how to remove all exclusive groups set on the design.

```
pt_shell> remove_si_aggressor_exclusion -all
1
```

SEE ALSO

```
report_delay_calculation(2)
report_noise_calculation(2)
report_si_aggressor_exclusion(2)
set_si_aggressor_exclusion(2)
set_si_delay_analysis(2)
set_si_noise_analysis(2)
si_analysis_logical_correlation_mode(3)
```

remove_si_delay_analysis

Removes the effect of the **set_si_delay_analysis** command.

SYNTAX

```
int remove_si_delay_analysis
   [-ignore_arrival inets]
   [-victims vnets]
   [-aggressors anets]
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-all]
```

Data Types

```
inets        list
vnets        list
anets        list
```

ARGUMENTS

-ignore_arrival *inets*

Removes the effect of using the **set_si_delay_analysis** command with its *-ignore_arrival* option. You cannot use this option with the *-victims* and *-aggressors* options.

-victims *vnets*

Removes the effect of using the **set_si_delay_analysis** command with its *-victims* option. When you use the *-victims* option with the *-aggressors* option, the command restores the pair-wise relationship. However, when the *-victims* option is used to remove the effect set by the **set_si_delay_analysis** command with the *-aggressors* option, it does not remove any exclusion on the nets. You cannot use the *-victims* option with the *-ignore_arrival* option.

-aggressors *anets*

Removes the effect of the **set_si_delay_analysis** command with its *-aggressors* option. When you use the *-aggressors* option with the *-victims* option, the command restores the pair-wise relationship. However, when the *-aggressors* option is used to remove the effect set by the **set_si_delay_analysis** command with the o*-victims* option, it does not remove any exclusion on the nets. You cannot use the *-aggressors* option with the *-ignore_arrival* option.

-rise

Removes the effect of the **set_si_delay_analysis -exclude -rise** command.

-fall

Removes the effect of the **set_si_delay_analysis -exclude -fall** command.

-min

Removes the effect of the **set_si_delay_analysis -exclude -min** command.

-max

Removes the effect of the **set_si_delay_analysis -exclude -max** command.

-all

Removes all the effects of the **set_si_delay_analysis** command on all the nets.

DESCRIPTION

Removes the effect of the **set_si_delay_analysis** command.

The **set_si_delay_analysis** command allows you to exclude nets from crosstalk analysis or setting some net as infinite window as infinite window as both aggressor and victim . The **remove_si_delay_analysis** command removes such overrides.

The **remove_si_delay_analysis** command returns a **1** if successful and a **0** if unsuccessful. If the remove command is used to remove a effect that has not been set, the warning message **XTALK-107** is issued.

To view the result of this command, use the **report_si_delay_analysis** command.

EXAMPLES

In the following example, all the nets named *CLK_NET_** are returned to their original state in crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET_*]
1

pt_shell> remove_si_delay_analysis -victims [get_nets CLK_NET_*]
1
```

However in the following examples, none of the nets named *REG_NET_** are returned to their original state in crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets REG_NET_*]
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets REG_NET_*]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2. (XTALK-107)
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets REG_NET_*]
-victims [get_nets CLK_NET]
1
```

If there is a VIC_NET with aggressors AGG_NET_* and you had set pairwise exclusion on the nets with the list of

its aggressors, the AGG_NET_* nets cannot be returned to their original state by using only *-aggressors* option. For example,

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets AGG_NET*]
1
```

In order to reset the pairwise exclusion on the nets VIC_NET and AGG_NET_*, the pairwise reset should be used as shown below

```
pt_shell> remove_si_delay_analysis -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1
```

To verify if the exclusion was removed on the nets, use the **report_si_delay_analysis** command with the *-excluded* option.

SEE ALSO

read_parasitics(2)
set_si_delay_analysis(2)
report_si_delay_analysis(2)
si_enable_analysis(3)

remove_si_delay_disable_statistical

Removes the effect of the **set_si_delay_disable_statistical** command.

SYNTAX

```
int remove_si_delay_disable_statistical
   dnets
```

Data Types

```
dnets      list
```

ARGUMENTS

dnets

A list of nets for which the effect of the **set_si_delay_disable_statistical** command is removed.

DESCRIPTION

Removes the effect of the **set_si_delay_disable_statistical** command.

The **set_si_delay_disable_statistical** command allows you to disable the statistical analysis for the *dnets* option if selected in composite aggressor group for crosstalk analysis.

The **remove_si_delay_disable_statistical** command removes such effect.

EXAMPLES

The following example shows how to put net back into statistical analysis when considered as a composite aggressor.

```
pt_shell> remove_si_delay_disable_statistical [get_nets LOGIC1]
1
```

SEE ALSO

```
set_si_delay_disable_statistical(2)
report_si_delay_analysis(2)
```

remove_si_noise_analysis

Removes the effect of the **set_si_noise_analysis** command.

SYNTAX

```
int remove_si_noise_analysis
   [-ignore_arrival inets]
   [-victims vnets]
   [-aggressors anets]
   [-above]
   [-below]
   [-low]
   [-high]
   [-all]
```

Data Types

inets list *vnets* list *anets* list

ARGUMENTS

-ignore_arrival *inets*

Removes the effect of using the **set_si_noise_analysis** command with its *-ignore_arrival* option. You cannot use this option with the *-victims* or *-aggressors* option.

-victims *vnets*

Removes the effect of using the **set_si_noise_analysis** command with its *-victims* option. When the *-victims* option is used to remove the effect set by the **set_si_noise_analysis** command with the *-aggressors* option, it does not remove any exclusion on the nets. However, when you use the *-victims* option with *-aggressors* option, the command restores the pair-wise relationship. You cannot use the *-victims* option with the **I-ignore_arrival** option.

-aggressors *anets*

Removes the effect of the **set_si_noise_analysis** command with its *-aggressors* option. When the *-aggressors* option is used to remove the effect set by the **set_si_noise_analysis** command with the *-victims* option, it does not remove any exclusion on the nets. However, when you use the *-aggressors* option with the *-victims* options, the command restores the pair-wise relationship. You cannot use the *-aggressors* option with the *-ignore_arrival* option.

-above

Removes the effect of the **set_si_noise_analysis -exclude -above** command.

-below

Removes the effect of the **set_si_noise_analysis -exclude -below** command.

-low

Removes the effect of the **set_si_noise_analysis -exclude -low** command.

-high

Removes the effect of the **set_si_noise_analysis -exclude -high** command.

-all

Removes all the effects of the **set_si_noise_analysis** command on all the nets.

DESCRIPTION

Removes the effect of the **set_si_noise_analysis** command.

The **set_si_noise_analysis** command allows you to exclude nets from noise analysis, or set some net as infinite window as aggressor. The **remove_si_noise_analysis** command removes such overrides.

The **remove_si_noise_analysis** command returns a **1** if successful and a **0** if unsuccessful. If remove command is used to remove a effect that has not been set, the warning message XTALK-107 is issued.

EXAMPLES

In the following example, all the nets named *CLK_NET_** are returned to their original state in noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET]
1

pt_shell> remove_si_noise_analysis -victims [get_nets CLK_NET*]
1
```

However in the following example, none of the nets named *REG_NET_** are returned to their original state in noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets REG_NET_*]
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets REG_NET_*]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2. (XTALK-107)
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets REG_NET_*]
-victims [get_nets CLK_NET]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1
CLK_NET. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2
CLK_NET. (XTALK-107)
1
```

If there is a VIC_NET with aggressors AGG_NET_* and you had set pairwise exclusion on the nets with the list of its aggressors, the AGG_NET_* nets cannot be returned to their original state in noise analysis. For example,

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets VIC_NET]
-aggressors [get_attribute -class net [get_nets VIC_NET] aggressors]
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets AGG_NET*]
Warning: Cannot remove an effect that was not set on net(s)
AGG_NET_1. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s)
AGG_NET_2. (XTALK-107)
1
```

In order to reset the pairwise exclusion on the nets VIC_NET and AGG_NET_*, the pairwise reset should be used as shown below

```
pt_shell> remove_si_noise_analysis -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1
```

To verify if the exclusion was removed on the nets, use the **report_si_noise_analysis** command with the *-excluded* option.

SEE ALSO

```
read_parasitics(2)
set_si_noise_analysis(2)
report_si_noise_analysis(2)
si_enable_analysis(3)
```

remove_si_noise_disable_statistical

Removes the effect of the **set_si_noise_disable_statistical** command.

SYNTAX

```
int remove_si_noise_disable_statistical
   dnets
```

Data Types

dnets list

ARGUMENTS

dnets

A list of nets for which the effect of the **set_si_noise_disable_statistical** command is removed.

DESCRIPTION

Removes the effect of the **set_si_noise_disable_statistical** command.

The **set_si_noise_disable_statistical** command allows you to disable the statistical analysis for the *dnets* option if selected in composite aggressor group for noise analysis.

The **remove_si_noise_disable_statistical** command removes such effect.

EXAMPLES

The following example shows how to put net back into statistical analysis when considered as a composite aggressor.

```
pt_shell> remove_si_noise_disable_statistical [get_nets LOGIC1]
1
```

SEE ALSO

```
set_si_noise_disable_statistical(2)
report_si_noise_analysis(2)
```

remove_steady_state_resistance

Removes steady state resistance for a library pin or port.

SYNTAX

```
int remove_steady_state_resistance
   [-above]
   [-below]
   [-low]
   [-high]
   object_list
```

Data Types

```
object_list        list
```

ARGUMENTS

-above

Removes the steady state resistance for above ground or power rail noise analysis region.

-below

Removes the steady state resistance for below ground or power rail noise analysis region.

-low

Removes the steady state resistance for ground rail noise.

-high

Removes the steady state resistance for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

DESCRIPTION

This command removes the steady state resistance information set by the **set_steady_state_resistance** command.

EXAMPLES

This example removes steady state resistance information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_steady_state_resistance -above lsi_10k/IV/A
```

SEE ALSO

set_steady_state_resistance(2)

remove_taps

Removes previously created tap points from the design.

SYNTAX

```
status remove_taps
   [taps]
```

Data Types

```
list taps
```

ARGUMENTS

taps

Specifies a list of taps to be removed. The taps can be specified as name patterns or collections of taps. When this argument is not specified, running the **remove_taps** command removes all of the taps from the design.

DESCRIPTION

The **remove_taps** command removes tap points that are previously created using the **create_taps** command.

To remove selective taps, use the selection mechanisms provided by the **get_taps** command. See the **get_taps** manpage for further details.

EXAMPLES

The following example shows how to remove all the supply pin taps that are of the type auto_pg:

```
pr_shell> remove_taps [get_taps -filter type==auto_pg]
```

The following example shows how to remove all the taps that are currently defined in the design:

```
pr_shell> remove_taps
```

SEE ALSO

```
create_taps(2)
get_taps(2)
report_taps(2)
write_taps(2)
```

remove_user_attribute

Removes a user attribute from an object.

SYNTAX

```
string remove_user_attribute
   [-quiet]
   [-class class_name]
   object_spec
   attr_name
```

Data Types

class_name	string
object_spec	list
attr_name	string

ARGUMENTS

-quiet

Does not report any messages.

-class *class_name*

If the *object_spec* option is a name, this is its class. Allowable values are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*.

object_spec

Shows objects from which to remove the attribute. Each element in the list is either a collection or a pattern that combines with the *class_name* option to find the objects.

attr_name

Provides the name of the attribute.

DESCRIPTION

The **remove_user_attribute** command removes attributes that you set with the **set_user_attribute** command.

You cannot remove application attributes with this command. Each application attribute that can be removed has a command dedicated to it. For example, the **fanout_load** attribute is removed with the **remove_fanout_load**

command.

EXAMPLES

This example defines an attribute 'X' for cells then sets the value on all cells in this level of the hierarchy. Finally, the example removes the attribute from one cell.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
pt_shell> remove_user_attribute [get_cells i1] X
Removed attribute 'X' from 'i1'
```

SEE ALSO

```
collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
set_user_attribute(2)
report_attribute(2)
```

remove_virtual_decap_cells

Removes virtual decap cells from the specified region.

SYNTAX

```
status remove_virtual_decap_cells
   -bounding_box rectangle
   [-all]
```

Data Types

```
list   rectangle
```

ARGUMENTS

-bounding_box *rectangle*

Specifies the rectangular region where the virtual decap cells are to be removed. The rectangle is specified as a list of four coordinates, {left bottom right top}. The coordinates need to be specified in the unit of micron.

-all

Removes all virtual decap cells in the design.

DESCRIPTION

The **remove_virtual_decap_cells** command removes the virtual decap cells from the specified region. When the -all option is specified, all virtual decap cells are removed.

EXAMPLE

The following example removes virtual decap cells from the specified region, {100 100 1000 1000}.

```
pr_shell> remove_virtual_decap_cells -region {100 100 1000 1000}
1
```

The following example removes all virtual decap cells in the design.

```
pr_shell> report_virtual_decap_cells -all
1
```

SEE ALSO

```
insert_virtual_decap_cells(2)
report_virtual_decap_cells(2)
calculate_rail_voltage(2)
```

remove_wire_load_min_block_size

Removes the minimum block size for automatic wire load selection.

SYNTAX

```
int remove_wire_load_min_block_size
```

ARGUMENTS

None.

DESCRIPTION

Removes the minimum block area for automatic wire load selection in the current design, set by the **set_wire_load_min_block_size** command.

EXAMPLES

The following example removes the previously-set minimum block size.

```
pt_shell> remove_wire_load_min_block_size
```

SEE ALSO

```
report_wire_load(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
```

remove_wire_load_model

Removes wire load model from designs, hierarchical cells, or ports.

SYNTAX

```
string remove_wire_load_model
   [object_list]
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

Lists ports, designs, or hierarchical cells. If this option is not specified, the wire load model is removed from the current instance, or from the current design if current instance is not set.

DESCRIPTION

Removes user-specified wire load model information on designs, ports, or hierarchical cells. The wire load model is used to calculate net capacitance, resistance, and area for designs, which are not placed and routed. You specify wire load models with the **set_wire_load_model** command, or by using automatic wire load selection.

To display wire load model settings for the current design or instance, use **report_wire_load**. To show wire load information for ports, use **report_port -wire_load**.

EXAMPLES

The following example removes wire load model settings from the current design and from all hierarchical cells in the design.

```
pt_shell> current_design TOP
pt_shell> remove_wire_load_model
pt_shell> remove_wire_load_model [get_cells -hier * -filter "is_hierarchical==true"]
```

The following example removes wire load model settings from port OUT2.

```
pt_shell> remove_wire_load_model [get_ports OUT2]
```

SEE ALSO

```
report_port(2)
report_wire_load(2)
set_wire_load_model(2)
set_wire_load_selection_group(2)
report_design(2)
auto_wire_load_selection(3)
```

remove_wire_load_selection_group

Removes wire load selection_group from current design.

SYNTAX

```
status remove_wire_load_selection_group
   [object_list]
```

Data Types

```
object_list        list
```

ARGUMENTS

object_list

Provides a list of hierarchical cells or designs. If you do not specify the *object_list* option, the wire load selection group is set on the current instance or on the current design if current instance is not set.

DESCRIPTION

This command removes the wire load selection group setting from the current design. Both min and max selection group information is removed. The wire load selection group is specified with **set_wire_load_selection_group**. For more information, see the **set_wire_load_selection_group** man page.

To show the wire load selection group information for a design, use the **report_design** command.

EXAMPLES

This example removes the wire load selection group setting from the current design.

```
pt_shell> remove_wire_load_selection_group
```

SEE ALSO

```
report_design(2)
set_wire_load_selection_group(2)
```

rename

Renames or deletes a command.

SYNTAX

```
string rename
   old_name
   new_name
```

ARGUMENTS

old_name

Specifies the current name of the command.

new_name

Specifies the new name of the command.

DESCRIPTION

Renames the *old_name* command so that it is now called *new_name*. If *new_name* is an empty string, then *old_name* is deleted. The *old_name* and *new_name* arguments may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The **rename** command returns an empty string as result.

Note that the **rename** command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: **rename** can have serious consequences if not used correctly. When using **rename** on anything other than a user-defined Tcl procedure, you will be warned. The **rename** command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like **puts**, **echo**, **open**, **close**, **source** and many others are often used within the application. Use **rename** with extreme care and at your own risk. Consider using **alias**, Tcl procedures, or a private namespace before using **rename**.

EXAMPLES

This example renames my_proc to my_proc2:

```
prompt> proc my_proc {} {echo "Hello"}
prompt> rename my_proc my_proc2
prompt> my_proc2
Hello
prompt> my_proc
Error: unknown command 'my_proc' (CMD-005).
```

SEE ALSO

define_proc_attributes(2)

rename_cell

Changes the name of a cell.

SYNTAX

```
int rename_cell
   cell
   new_name
```

Data Types

```
cell            list
new_name        string
```

ARGUMENTS

cell

Specifies one cell to be renamed.

new_name

Specifies the new name for *cell*.

DESCRIPTION

The **rename_cell** command changes the name of a cell. Either a cell name that matches a single cell, or a collection of one cell, is passed in as the *cell* argument.

The **rename_cell** command fails if any of the following are true:

- PrimeTime cannot find *cell*.
- PrimeTime finds multiple cells that match *cell*.
- PrimeTime finds a leaf cell matching *new_name*.
- *new_name* is not at the same level of hierarchy as *cell*.

The **write_changes** command records the **rename_cell** command.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the

design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following examples, a cell is renamed in the current instance.

```
pt_shell> rename_cell cellA cellB
Information: Renamed cell 'cellA' to 'cellB' in design 'top'. (NED-008)
1
```

In the following examples, a cell is renamed at a level below the current instance. Currently, H1 and H1/H2 are instances of designs that have multiple instances. Therefore, they are uniquified as part of the editing operation.

```
pt_shell> rename_cell H1/H2/u1 H1/H2/cellC
Uniquifying 'H1/H2' (low) as 'low_0'.
Uniquifying 'H1' (inter) as 'inter_0'.
Information: Renamed cell 'u1 to 'cellC' in 'top/H1/H2'. (NED-008)
1
```

SEE ALSO

```
rename_design(2)
rename_net(2)
write_changes(2)
```

rename_design

Change the name of a design.

SYNTAX

```
int rename_design
   design
   new_name
```

Data Types

```
design          list
new_name        string
```

ARGUMENTS

design

Specifies a design to be renamed. Only one design can be specified.

new_name

Specifies the new name for the *design* option.

DESCRIPTION

The **rename_design** command changes the name of a design. Either a design name which matches a single design, or a collection of one design, is passed in as the *design* argument. The command allows only simple design name changes for a single design. You cannot change the association of a design with its source file.

For the *design* to be renamed to *new_name*, there cannot be a design named *new_name* in the same file as *design* (see the Examples). The command may also fail if there are any instances of *design* in a linked design.

Renaming designs in PrimeTime should be done prior to any linking to avoid the case of instantiation conflicts.

Note that the **rename_design** command is not recorded for output with the **write_changes** command.

EXAMPLES

In the following example, design *top* is successfully renamed to *top2*:

```
pt_shell> list_designs
Design Registry:
 *  top            /home/designs/top.v:top
1
pt_shell> rename_design [get_designs top] top2
Renamed design 'top' to 'top2'
1
pt_shell> list_designs
Design Registry:
 *  top2           /home/designs/top.v:top2
1
```

In the following example, design *fbox* cannot be renamed *mbox* because there is already a design of that name in the same file. An attempt to rename it to *fbox2* fails because *fbox* is instantiated in a linked design.

```
pt_shell> list_designs
Design Registry:
    fbox           /home/designs/top.v:fbox
    mbox           /home/designs/top.v:mbox
 *L top            /home/designs/top.v:top
1
pt_shell> rename_design fbox mbox
Error: Could not rename design 'fbox' to 'mbox':
 design 'mbox' already exists in file
        '/home/ajs/designs/top.v' (NED-020)
0
pt_shell> rename_design fbox fbox2
Error: Could not rename design 'fbox' to 'fbox2':
 design 'fbox' is instantiated in design 'top' (NED-020)
0
```

SEE ALSO

```
list_designs(2)
link_design(2)
write_changes(2)
```

rename_integrity_result

Change the name of an existing integrity result.

SYNTAX

```
status rename_integrity_result
   old_result_name
   new_result_name
   [-replace]
```

Data Types

```
string old_result_name
string new_result_name
```

ARGUMENTS

old_result_name

Name of the existing integrity result to be renamed.

new_result_name

New name of the result after the rename operation.

-replace

Allows you to overwrite the previous *new_result_name* result if one exists. By default existing results cannot be overwritten.

DESCRIPTION

Renames an existing integrity result. Integrity results are generated by **check_supply_net_integrity** command and are assigned names by that commands, either by an option to the command or by automatic assignment. The **rename_integrity_result** command allows you to reassign the name of the result at any time during the session. You may want to assign more meaningful names to integrity results if you are switching back and forth between multiple results while analyzing them.

By default the **rename_integrity_result** command does not overwrite an existing integrity result: if the *new_result_name* is already assigned to an integrity result then the command does not perform the renaming. You can use the *-replace* option to force the command to overwrite an existing result. Note that when you do this the previous result is deleted and cannot be recovered. If the current integrity result (see **current_integrity_result**) is renamed, then the current integrity result becomes the *new_result_name*

integrity result.

EXAMPLES

The following commands rename the integrity result MY_INTEGRITY_RESULT_1 to INTEGRITY_RESULT_3:

```
pr_shell> report_integrity_result
****************************************
Report : integrity check results
Design : Micro
Version: E-2010.12
Date   : Mon Jan 17 17:52:11 2011
****************************************
   MY_INTEGRITY_RESULT_1
   INTEGRITY_RESULT_2
1


pr_shell> rename_integrity_result MY_INTEGRITY_RESULT_1 INTEGRITY_RESULT_3
1

pr_shell> report_integrity_result
****************************************
Report : integrity check results
Design : Micro
Version: E-2010.12
Date   : Mon Jan 17 17:52:55 2011
****************************************
   INTEGRITY_RESULT_3
   INTEGRITY_RESULT_2
1
```

The following commands overwrite the existing INTEGRITY_RESULT_3 with a different result:

```
pr_shell> rename_integrity_result INTEGRITY_RESULT_2 INTEGRITY_RESULT_3 -replace
1

pr_shell> report_integrity_result
****************************************
Report : integrity check results
Design : Micro
Version: E-2010.12
Date   : Mon Jan 17 17:52:55 2011
****************************************
   INTEGRITY_RESULT_3
1

pr_shell> current_integrity_result
INTEGRITY_RESULT_3
1
```

SEE ALSO

current_integrity_result(2)
report_integrity_results(2)

```
remove_integrity_result(2)
pr_work_dir(3)
```

rename_mw_lib

Renames a Milkyway library.

SYNTAX

```
status rename_mw_lib
   -from lib_name
   -to lib_name
```

Data Types

```
lib_name     string
```

ARGUMENTS

-from *lib_name*

Specifies the name of the Milkyway library that is to be renamed. The specified Milkyway library must not be open in the current session.

-to *lib_name*

Specifies the new name of the Milkyway library.

DESCRIPTION

This command renames a Milkyway library.

If this library is a reference library of another main library, you may need to use the **set_mw_lib_reference** command to update reference library information in the main library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example changes the name of the library from *access05* to *access06*:

```
prompt> rename_mw_lib -from access05 -to access06
```

SEE ALSO

copy_mw_lib(2)

rename_net

Changes the name of a net.

SYNTAX

```
status rename_net
   net
   new_name
```

Data Types

```
net          list
new_name     string
```

ARGUMENTS

net

Specifies a net to be renamed. Only one net can be specified.

new_name

Specifies the new name for *net*.

DESCRIPTION

The **rename_net** command changes the name of a net. Either a net name that matches a single net or a collection of one net is passed in as the *net* argument.

The **rename_net** command fails if any of the following are true:

- PrimeTime cannot find *net*.
- PrimeTime finds multiple nets that match *net*.
- PrimeTime finds a net, port, or hierarchical pin matching *new_name* in the same hierarchical block as *net*.
- *new_name* is not at the same level of hierarchy as *net*.

The **rename_net** command is recorded for output with the **write_changes** command.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the

design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is re-linked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following examples, a net is renamed in the current instance.

```
pt_shell> rename_net netA netB
Information: Renamed net netA' to 'netB' in design 'top'. (NED-009)
1
```

In the following examples, a net is renamed at a level below the current instance. Currently, H1 and H1/H2 are instances of designs that have multiple instances. Therefore, they are uniquified as part of the editing operation.

```
pt_shell> rename_net H1/H2/w1 H1/H2/netC
Uniquifying H1/H2 (low) as 'low_0'.
Uniquifying 'H1' (inter) as 'inter_0'.
Information: Renamed net 'w1' to 'netC' in 'top/H1/H2'. (NED-009)
1
```

SEE ALSO

```
rename_cell(2)
rename_design(2)
write_changes(2)
```

rename_rail_result

Changes the name of an existing rail result.

SYNTAX

```
status rename_rail_result
   name
   new_name
   [-replace]
```

Data Types

```
string name
string new_name
```

ARGUMENTS

name

Specifies the name of the existing rail result to be changed.

new_name

Specifies the new name to be assigned.

-replace

Overwrites an existing result if it is also named *new_name.* By default, the existing results cannot be overwritten.

DESCRIPTION

The command renames an existing rail result. Rail results are generated by certain rail analysis commands, such as **calculate_rail_voltage** or **calculate_minimum_path_resistances**, and are named by those commands, either by an option to the command or by automatic assignment. The **rename_rail_result** command allows you to reassign the name of the result at any time during the session. For example, you may want to assign more meaningful names to the rail results if you are switching back and forth between multiple results while analyzing them.

By default, the **rename_rail_result** command does not overwrite an existing rail result. If the *new_name* is already assigned to an existing rail result, the command does not perform the renaming. You can use the *-replace* option to force the command to overwrite an existing result. Note that when you do this the previous result is deleted and cannot be recovered. If the name to be renamed is the one assigned by the **current_rail_result**

command, the current rail result becomes the *new_name* rail result.

EXAMPLES

The following command renames the rail result *VDROP_RAIL_RESULT_2* to *VDROP_RAIL_RESULT_1*:

```
pr_shell> current_rail_result
VDROP_RAIL_RESULT_2

pr_shell> report_rail_result -verbose

****************************************
Report : rail analysis results
Design : MyDesign
Version: E-2010.12
Date   : Thu Aug 26 14:47:16 2010
****************************************

   VDROP_RAIL_RESULT_2
      Nets in this result:
      VDD
      VDDG
      VDDGS
      VDDX
      VDDXS
      VDDY
      VDDYS
      VSS

1


pr_shell> rename_rail_result VDROP_RAIL_RESULT_2 VDROP_RAIL_RESULT_1 -replace
1

pr_shell> current_rail_result
VDROP_RAIL_RESULT_1

pr_shell> report_rail_result -verbose

****************************************
Report : rail analysis results
Design : MyDesign
Version: E-2010.12
Date   : Thu Aug 26 14:47:16 2010
****************************************

   VDROP_RAIL_RESULT_1
      Nets in this result:
      VDD
      VDDG
      VDDGS
      VDDX
      VDDXS
      VDDY
      VDDYS
      VSS


1
```

SEE ALSO

```
calculate_minimum_path_resistances(2)
calculate_rail_voltage(2)
current_rail_result(2)
remove_rail_result(2)
report_rail_results(2)
pr_work_dir(3)
```

report_activity_file_check

Reads an activity file without annotating the switching activity, and reports the proceedings of the mapping. proceeds.

SYNTAX

```
string report_activity_file_check
   [-rtl]
   [-format SAIF | VCD]
   [-pipe_exec pipe]
   [-path prefix]
   -strip_path strip_path
   [-nosplit]
   [-find pattern]
   file_name
```

Data Types

```
pipe                 string
prefix               string
strip_path           string
pattern              string
file_name            string
```

ARGUMENTS

-rtl

Specifies that the name mapping method must be applied during the *time_based* power analysis mode. As the name mapping method is always applied in the averaged power analysis mode, the **-rtl** option is disabled in the averaged power analysis mode. For more information about specifying name mapping, see the **set_rtl_to_gate_name** man page.

-format SAIF | VCD

Specifies whether the file referenced by the *file_name* argument is a VCD file or a SAIF file. The *format* argument is not necessary if the format of the *file_name* can be recognized by the file name extension.

-pipe_exec *pipe*

Specifies a shell command used to generate the VCD file using the *file_name* option. This option invokes the shell command and directly pipes the output VCD file to PrimeTime PX. There is no VCD disk file generated at all. You cannot use this option if the value specified with the *format* argument is SAIF.

-path *prefix*

Specifies a relative path from the current design to the hierarchical low- level design for which the VCD file has

been created. By default, the absolute path names are used. Use this option if the VCD file refers to an object in a hierarchy. Do not use this option if the VCD file refers to an absolute path.

-strip_path ***strip_path***

Specifies a path prefix that must be stripped from all the object names read from the VCD or SAIF file. This option is applied to strip the testbench or instance path from the RTL VCD or SAIF file.

-nosplit

Prevents line splitting if column overflows.

-find ***pattern***

Specifies to only report on activity file objects that match the *pattern* option. By only reporting on some activity file objects, the report can be much shorter and more readable. The pattern is a glob, so a pattern such as "A/B*" matches match "A/B1", "A/B2", "A/B/C/D". Multiple globs can be included with commas {A/B*, C/*/Q}. Only the activity file objects are checked to see if the pattern matches. Any activity file object not matching the pattern is not reported.

file_name

Specifies the name of the switching activity file to be read. The specified file can be a SAIF, VCD, VPD, or FSDB file and can be of compressed format. The file name extension can be used to determine the file type. The file names with the ".vcd", ".vcd.gz", ".vcd.Z", ".vcd.bz2", ".vpd" or ".fsdb" extensions are converted to RTL VCD files and read as RTL VCD files; whereas the file names with the ".saif", ".saif.Z", ".saif.gz", or ".saif.bz2" extensions are converted to RTL SAIF files and read as RTL SAIF files.

DESCRIPTION

The **report_activity_file_check** command helps debug the switching activity annotation from simulation. The command processes the activity file in the same way as the **read_saif** or **read_vcd** command would, except that annotation is not applied to the design. This command reports the proceedings of the name mapping. For each hierarchical module (in VCD) or instance (in SAIF), the corresponding hierarchical cell in the design is displayed, if found. For each signal (in VCD) or net (in SAIF), the corresponding net or pin in the design is shown, if any. The exact method (or combination of methods) used to arrive at the mapping is also reported. In addition, this command also reports information about any logical inversion between the signal in VCD (or net in SAIF) and the net or pin in the design, this information is displayed.

Using the **report_activity_file_check** command, it is possible to determine where in the design a particular signal in a VCD would be annotated by the **read_vcd** command. To do this, run the **report_activity_file_check** command with the *-find* option to output only the results matching the signal in question.

When you set the **power_analysis_mode** variable to *time_based*, you cannot use the SAIF files with the **report_activity_file_check** command.

EXAMPLES

The following example reports the mapping a SAIF file to the design. Note that nothing was reported for the signal Ud SAIF file.

```
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut -path top
****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
-------------------------------------------------------------------------------------------

      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
-------------------------------------------------------------------------------------------
File Type: saif
-------------------------------------------------------------------------------------------
Activity    Activity                        Design     Design          Name      Attributes
File        File                            Object     Object          Mapping
Object Type Name                            Type       Name            Method
-------------------------------------------------------------------------------------------
instance    (strip_path matches)            Design
net         Ua                              Net        Ua/Q            e
net         Ub                              Net        Ub/Q            e
net         Ud                              Net        None Found
net         clk                             Net        clk             e
net         in                              Net        in              e
-------------------------------------------------------------------------------------------
```

The following example reports the mapping a SAIF file to the design after adding a name-mapping command to fix the missing switching activity annotation to signal Ud.

```
pt_shell> set_rtl_to_gate_name -rtl {Ud} -gate {Uc/Q}
1
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut -path top


****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
-------------------------------------------------------------------------------------------
      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
```

```
       v - Inverted
-------------------------------------------------------------------------------------------
File Type: saif
-------------------------------------------------------------------------------------------
Activity    Activity                        Design     Design        Name      Attributes
File        File                            Object     Object        Mapping
Object Type Name                            Type       Name          Method
-------------------------------------------------------------------------------------------
instance    (strip_path matches)            Design
net         Ua                              Net        Ua/Q         e
net         Ub                              Net        Ub/Q         e
net         Ud                              Net        Uc/Q    m
net         clk                             Net        clk          e
net         in                              Net        in           e
-------------------------------------------------------------------------------------------
```

The following example reports the mapping a SAIF file to the design, and limits the output by using the *-find* option.

```
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut -path top -find U*

****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
-------------------------------------------------------------------------

      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
-----------------------------------------------------------------------------
File Type: saif
-----------------------------------------------------------------------------
Activity    Activity   Design     Design      Name      Attributes
File        File       Object     Object      Mapping
Object Type Name       Type       Name        Method
-----------------------------------------------------------------------------
net         Ua         Net        Ua/Q        e
net         Ub         Net        Ub/Q        e
net         Ud         Net        Uc/Q        m
-----------------------------------------------------------------------------
```

SEE ALSO

```
set_rtl_to_gate_name(2)
read_vcd(2)
read_saif(2)
```

```
power_analysis_mode(3)
```

report_activity_waveforms

Reports on activity analysis of VCD.

SYNTAX

```
int report_activity_waveforms
   [-nosplit]
```

ARGUMENTS

-nosplit

Specifies that lines with overflow should not be split. This can be useful when the output is read by a script.

DESCRIPTION

After running the **write_activity_waveforms** command, you can use the **report_activity_waveforms** command to summarize the results and give the peak activity times for each block in the VCD file analyzed.

EXAMPLES

This example shows the form of the report.

```
pt_shell> write_activity_waveforms -output filename1 -vcd my_vcd.vcd -interval 10
pt_shell> report_activity_waveforms

****************************************
Report : Time-Based Switching Activity
Activity File: my_vcd.vcd
Version: B-2008.06-Dev-DEV
Date   : Wed Apr  9 17:54:06 2008
****************************************

Block                            # Signals      Peak           Peak           Peak
Name                                            Interval       Interval       Toggle
                                                Start          End            Rate
-------------------------------------------------------------------------------------------------
TOP_level_of_my_design           1383           0              2000           0.000192
 Second_level_of_my_design       1335           0              2000           0.000199
-------------------------------------------------------------------------------------------------
```

SEE ALSO

`write_activity_waveforms(2)`

report_alternative_lib_cells

Generates a report that contains data to aid in the selection of alternative library cells for a cell in the current design.

SYNTAX

```
string report_alternative_lib_cells
   [-current_library]
   [-libraries lib_spec]
   [-delay_type delay_type]
   [-all_pins]
   [-weighted_design_cost]
   [-total_design_cost]
   [-significant_digits digits]
   [-nosplit]
   cell
```

Data Types

cell	string
lib_spec	list
delay_type	string
digits	int

ARGUMENTS

-current_library

If you specify this option, PrimeTime searches for library cells in the cell's current library only. You cannot specify this option with the *-libraries* option.

-libraries *lib_spec*

Specifies a list of libraries from which to select alternative library cells. The default is to select from all libraries in the link path. The *lib_spec* can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the **get_libs** command. You cannot specify this option with the *-current_library* option.

-delay_type *delay_type*

Specifies the delay type to be reported for slack values. Allowed values are *max* (the default), *min, min_max, max_rise, max_fall, min_rise*, or *min_fall*. For the *min* and *max* values, the worst of rise and fall values is reported.

-all_pins

Indicates that the report is to include all pins. By default, the report includes the worst of the output pins.

-weighted_design_cost

Indicates that the report is to contain the weighted design cost (negative slack). The value is calculated as a weighted sum over the weighted cost values calculated for all path groups. The cost values reported are the same as those reported by the **report_constraints** command for minimum delay (costs as a result of hold constraints) and maximum delay (costs as a result of setup constraints).

The minimum delay value is calculated as follows:

```
weighted min_delay/hold cost = summation over all path groups
( group weight * sum of all non-zero hold costs in group )
```

The maximum delay value is calculated as follows:

```
weighted max_delay/setup cost = summation over all path groups
( group weight * worst setup cost in group )
```

Maximum cost values are reported when *-delay_type max*, *max_rise*, or *max_fall* is specified, or when the default (*max*) is accepted. Minimum cost values are reported when *-delay_type min*, *min_rise*, or *min_fall* is specified. If *min_max* is specified, both minimum and maximum values are reported.

-total_design_cost

Indicates that the report is to contain the total design cost (negative slack). This option is similar to the *-weighted_design_cost* option, except that the total design cost can change in cases when the weighted maximum delay (setup) cost does not. Hence, this value can give some indication of the overall improvement in the design timing when the change does not improve the worst setup timing violation.

The total design cost is also computed as a weighted sum of the total costs computed for each path group. Only positive cost values are included in the calculation. The value is calculated as follows:

```
total design cost = summation over all path groups
( group weight * group total cost )

group total cost = summation over all group endpoints
having positive cost ( cost value )
```

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported.

Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

cell

Specifies a cell for which a collection of alternative library cells is required.

DESCRIPTION

The **report_alternative_lib_cells** command generates a report that contains data to aid in the selection of

alternative library cells for a cell in the current design. A report is produced that provides the slack values that would be obtained by replacing the library cell referenced by the cell with each alternative library cell that could be used to size the cell. A size cell operation is actually performed on the cell using each alternative library cell, followed by a timing analysis, and the slack values are obtained for each alternative. For each alternative library cell, the report lists the worst slack over all cell output pins.

You can refine the type of reported slack values using the *-delay_type* option. Use the *-all_pins* option to report the values for all cell pins (inputs, outputs, and inouts). Alternative library cells must satisfy the criteria used by the **size_cell** command to be considered valid alternatives.

If the *-current_library* option is specified, then PrimeTime searches for library cells in the cell's current library. If the *-libraries* option is specified, PrimeTime searches for library cells in the libraries contained only within the *lib_spec*.

Alternatively, if neither of the above options are specified, PrimeTime searches for the library cell in the following sources in this order:

1. The cell's current library.
2. The **link_path_per_instance** variable.
3. The **link_path** variable, if no **link_path_per_instance** specification pertains to the cell.

To create a collection of library cells that can be used to "size" a specified cell, use the **get_alternative_lib_cells** command.

EXAMPLES

The following example illustrates generating a report of the alternatives library cells. The alternatives are taken from all libraries in the link path (the default). The default delay type, *max*, is used.

```
pt_shell> report_alternative_lib_cells AND2_0

****************************************
Report : alternative_lib_cells
        mux0
        -delay_type max
Design : swap0/0
Version: 2000.05-SI1
Date   : Fri Dec 17 14:37:22 1999
****************************************

Alternative        Slack
Library Cells
--------------------------------------------------------------------------------
AND2_2             -1.34(r)
AND2_1             -1.78(f)
AND2_0 *           -2.34(r)
AND2_3             -3.01(f)

*: The original library cell, not an alternative.
```

The following example generates a report that includes minimum and maximum slacks.

```
pt_shell> report_alternative_lib_cells AND2_0 -delay_type min_max

****************************************
Report : alternative_lib_cells
```

```
         mux0
         -delay_type min_max
Design : swap0/0
Version: 2000.05-SI1
Date   : Fri Dec 17 14:37:23 1999
****************************************

Alternative                  Slack (min:max)
Library Cells
--------------------------------------------------------------------------------
AND2_1                       1.34:8.00(r)
AND2_2                      -1.01:-1.00(f)
AND2_0 *                     2.00:-2.34(r)
AND2_3                       3.34:-3.01(f)

*: The original library cell, not an alternative.
```

The following example generates a report that contains the weighted and total design costs. The alternative cells are to be chosen from all libraries in memory. The example illustrates that the weighted design cost is not affected by any of the alternatives; however, the total cost of the design can be improved by using cells AND2_2 or AND2_1 over the current cell AND2_0.

```
pt_shell> set libs_in_mem [get_libs *]
pt_shell> report_alternative_lib_cells AND2_0 -libraries $libs_in_mem
              -delay_type max -weighted_design_cost -total_design_cost
****************************************
Report : alternative_lib_cells
        mux0
        -delay_type max
Design : swap0/0
Version: 2000.05-SI1
Date   : Fri Dec 17 14:37:22 1999
****************************************

Alternative        Slack        Weighted            Total Design Cost
Library Cells                   max_delay/setup
                                Cost
--------------------------------------------------------------------------------
AND2_2             1.01(r)      4.03                35.67
AND2_1            -1.34(f)      4.03                34.34
AND2_0 *          -2.34(r)      4.03                36.78
AND2_3            -3.01(f)      4.03                37.88

*: The original library cell, not an alternative.
```

SEE ALSO

```
get_alternative_lib_cells(2)
get_libs(2)
set_min_library(2)
size_cell(2)
link_path(3)
```

report_analysis_coverage

Generates a report about the coverage of timing checks.

SYNTAX

```
status report_analysis_coverage
   [-status_details status_list]
   [-check_type check_type_list]
   [-exclude_untested untested_reason_list]
   [-sort_by sort_method]
   [-significant_digits digits]
   [-nosplit]
   [-pre_commands pre_command_string]
   [-post_commands post_command_string]
```

Data Types

status_list	list
check_type_list	list
untested_reason_list	list
sort_method	string
digits	integer
pre_command_string	string
post_command_string	string

ARGUMENTS

-status_details *status_list*

Specifies a space-delimited list of status names about which to show detail in the report. The allowed values are **untested**, **violated**, and **met**. By default, only summary information is shown. Use this option to see information about individual timing checks.

-check_type *check_type_list*

Specifies a list of check types to include in the report. The allowed values are **setup**, **hold**, **recovery**, **removal**, **min_period**, **min_pulse_width**, **clock_separation**, **data_separation**, **max_skew**, **clock_gating_setup**, **clock_gating_hold**, **out_setup**, **out_hold**, and **nochange**.

Check types **out_setup** and **out_hold** refer to the output setup and output hold constraints generated by the **set_output_delay** command.

-exclude_untested *untested_reason_list*

Specifies a space-delimited list of reasons to exclude an untested check in the report. A check classified as untested is excluded from the report for any of the reasons specified as values to this argument. Therefore, the coverage statistics are unaffected by these excluded constraints. The following values are allowed:

- **constant_disabled** - Paths to this check are disabled because of case analysis or a logic constant propagated through the design (for example, caused by a signal tied high or low).
- **mode_disabled** - A timing constraint is disabled because it requires a mode to be selected in mode analysis, and that mode is not selected.
- **user_disabled** - The timing check is explicitly disabled by the user.
- **no_paths** - The timing check had no paths found to it and as a result there were no arrival times.
- **false_paths** - All paths were false to a constrained pin.
- **no_endpoint_clock** - The timing check has no destination clock signal to latch the data.
- **no_startpoint_clock** - The timing check has no clock that launches the data at a startpoint latch.
- **no_constrained_clock** - There is no constrained clock for skew or clock separation checks.
- **no_ref_clock** - There is no reference clock for skew or clock separation checks.
- **no_clock** - A minimum pulse width or period width check has no clock.
- **unknown** - The reason is not listed above.

-sort_by ***sort_method***

Specifies the sorting method for the output of the detailed list. The following values are allowed:

- **slack** (the default) - Sorts by the slack (untested is treated as negative infinity slack), followed by the pin name and check type.
- **name** - Sorts alphabetically by the name of the constrained pin, followed by the slack and check type.
- **check_type** or **check** - Sorts alphabetically by the check type, followed by the slack and pin name.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported. The allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default is 2. Use this option if you want to override the default.

-nosplit

Prevents line-splitting when columns overflow.

-pre_commands ***pre_command_string***

This option is available only if you invoke PrimeTime with the **-multi_scenario** option. This option allows you to specify a string of commands to be executed in the slave context before the execution of the merged reporting command. Commands must be grouped using the ";" character. The maximum size of a command is 1000 characters.

-post_commands ***post_command_string***

This option is available only if you invoke PrimeTime with the **-multi_scenario** option. This option allows you to specify a string of commands to be executed in the slave context after the execution of the merged reporting command. Commands are grouped using the ";" character. The maximum size of a command is 1000 characters.

DESCRIPTION

Generates a report showing information about the coverage of timing checks in the current design, or current instance if it is defined. The default report is a summary of checks by type. For each type of check (for example, **setup**), the report shows the number and percentage of checks meeting and violating constraints, and those that are untested; the report does not show check types for which there are no checks. The report does not show unconstrained output ports; that is, those that have no output delay or max_delay or min_delay; however, the report does show constrained output ports.

For more information about the individual timing checks, use the **-status_details** option, which shows all checks with the corresponding status. Untested checks show information about the reason why they are untested, if the reason can be determined.

By default, the coverage statistics include constraints that have been disabled. Constraints could be disabled because of constant propagation (for logic constants in the design and case analysis), modes that are not enabled, or explicit disabling of timing arcs by the user. Such disabled constraints appear as untested in the coverage statistics. If you want to exclude some of these untested constraints from the coverage statistics, use the **-exclude_untested** option.

Note that the command reports primarily library defined timing checks, in addition to **output_setup** and **output_hold**, which can be set by the **set_output_delay** command. User-defined constraints such as those set by **set_max_delay**/**set_min_delay** commands are not reported by **report_analysis_coverage**.

In the distributed analysis mode, only the **-full_design** option is supported at the master process. If you do not specify any option at the master, the **-full_design** options is enabled by default. All other options are supported with the command when executed on remote workers as part of the distributed command **remote_execute -partitions**.

EXAMPLES

The following example shows the summary report.

```
pt_shell> report_analysis_coverage
****************************************
Report : analysis_coverage
Design : counter
 ...
****************************************

Type of Check         Total      Met         Violated    Untested
-----------------------------------------------------------------
setup                  5         0 (  0%)    3 ( 60%)    2 ( 40%)
hold                   5         3 ( 60%)    0 (  0%)    2 ( 40%)
-----------------------------------------------------------------
All Checks            10         3 ( 30%)    3 ( 30%)    4 ( 40%)
```

The following example shows the detailed report of untested setup checks.

```
pt_shell> report_analysis_coverage -status_details {untested} -check_type {setup}
****************************************
Report : analysis_coverage
        -status_details {untested }
        -sort_by slack
```

```
        -check_type {setup }
Design : counter
 ...
****************************************

Type of Check         Total              Met         Violated         Untested
-------------------------------------------------------------------------------
setup                     5         0 (  0%)         3 ( 60%)         2 ( 40%)
-------------------------------------------------------------------------------
All Checks                5         0 (  0%)         3 ( 60%)         2 ( 40%)


Constrained                            Related   Check
Pin                                    Pin       Type         Slack  Reason
-------------------------------------------------------------------------------
ffd/CR                                 CP        setup     untested  no_clock
ffd/D                                  CP        setup     untested  no_clock
```

The following example shows generation of a report that includes details of untested and violated checks, excludes timing checks if they are untested because of constant propagation, sorts by slack, and shows the **setup** and **out_setup** checks.

```
pt_shell> report_analysis_coverage -check_type {out_setup setup} \
          -exclude_untested {constant_disabled} -status_details {violated untested}
****************************************
Report : analysis_coverage
        -status_details {untested violated }
        -exclude_untested {constant_disabled}
        -sort_by slack
        -check_type {setup out_setup}
Design : seq_case
 ...
****************************************

Type of Check         Total              Met         Violated         Untested
-------------------------------------------------------------------------------
setup                     6         5 ( 83%)         1 ( 17%)         0 (  0%)
out_setup                 5         4 ( 80%)         0 (  0%)         1 ( 20%)
-------------------------------------------------------------------------------
All Checks               11         9 ( 82%)         1 (  9%)         1 (  9%)


Constrained                            Related   Check
Pin                                    Pin       Type         Slack  Reason
-------------------------------------------------------------------------------
Q2                                               out_setup untested  no_paths
ff4/TI                                 CP        setup        -2.30
```

SEE ALSO

check_timing(2)
current_instance(2)
report_constraint(2)
report_timing(2)
report_default_significant_digits(3)

report_annotated_check

Reports back-annotated timing checks.

SYNTAX

```
string report_annotated_check
   [-setup]
   [-hold]
   [-recovery]
   [-removal]
   [-nochange]
   [-width]
   [-period]
   [-max_skew]
   [-clock_separation]
   [-max_line num]
   [-list_annotated]
   [-list_not_annotated]
   [-list_lib_annotated]
   [-constant_arcs]
```

Data Types

```
num        int
```

ARGUMENTS

-setup

Reports setup timing checks.

-hold

Reports hold timing checks.

-recovery

Reports recovery timing checks.

-removal

Reports removal timing checks.

-nochange

Reports nochange timing checks.

-width

Reports minimum pulse width timing checks.

-period

Reports minimum period timing checks.

-max_skew

Reports maximum skew timing checks.

-clock_separation

Reports clock separation timing checks.

-max_line *num*

Provides maximum number of line for the *-list_** options.

-list_annotated

Lists timing arcs that are back-annotated. Use this option if no annotated checks are expected to identify the specified annotated checks.

-list_not_annotated

Lists timing arcs that are not back-annotated. Use this option if some annotated checks are missing to identify the missing annotated checks.

-list_lib_annotated

Lists library timing arcs that are back-annotated.

-constant_arcs

Keeps a separate count for the arcs that are disabled due to logic constants (but not case_analysis). With this option, the *-list_annotated* or *-list_not_annotated* option does not list the arc disabled due to logic constants. The total count at the end of the table includes the constant_arcs count.

DESCRIPTION

Provides a summary report of how many cell timing checks are annotated in the current design. The purpose of this command is to check if all timing checks of the design are annotated after reading an SDF file. You can use the *-list_not_annotated* option to identify which check timing arcs are not annotated.

EXAMPLES

The following are examples of annotated reports.

```
pt_shell> report_annotated_check

******************************
report: annotated_check
******************************
```

```
                              |           |           |    NOT    |
                              |   Total   | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell setup arcs               |         2 |         0 |         2 |
cell hold arcs                |         2 |         0 |         2 |
cell recovery arcs            |         2 |         0 |         2 |
cell removal arcs             |         2 |         0 |         2 |
cell nochange arcs            |         2 |         0 |         2 |
cell min pulse width arcs     |         2 |         0 |         2 |
cell min period arcs          |         2 |         0 |         2 |
cell max skew arcs            |         2 |         0 |         2 |
cell clock separation arcs    |         2 |         0 |         2 |
-----------------------------+-----------+-----------+-----------+
                              |        18 |         0 |        18 |

pt_shell> report_annotated_check -constant_arcs

******************************
report: annotated_check
******************************

                              |           |           |    NOT    |
                              |   Total   | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell setup arcs               |         2 |         0 |         1 |
constant arcs                 |           |         0 |         1 |
cell hold arcs                |         2 |         0 |         1 |
constant arcs                 |           |         0 |         1 |
cell recovery arcs            |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell removal arcs             |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell nochange arcs            |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell min pulse width arcs     |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell min period arcs          |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell max skew arcs            |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
cell clock separation arcs    |         2 |         0 |         2 |
constant arcs                 |           |         0 |         0 |
-----------------------------+-----------+-----------+-----------+
                              |        18 |         0 |        18 |
```

SEE ALSO

read_sdf(2)
set_annotated_check(2)
remove_annotated_check(2)
reset_design(2)

report_annotated_delay

Reports back-annotated delays.

SYNTAX

```
string report_annotated_delay
   [-cell]
   [-net]
   [-from_in_ports]
   [-to_out_ports]
   [-max_line num]
   [-list_annotated]
   [-list_not_annotated]
   [-constant_arcs]
```

Data Types

```
num     integer
```

ARGUMENTS

-cell

Reports all data annotated on cells.

-net

Reports all data annotated on nets.

-from_in_ports

Includes the nets that start from input ports. By default, nets that start at input ports are not included in the list of nets reported for annotated delays. The reason is that SDF format specifies the nets starting at input ports of the design are not part of the SDF file because the delay of this net is dependent on the external environment.

-to_out_ports

Includes the nets that end at output ports. By default, nets that end at output ports are not included in the list of nets reported for annotated delays. The reason is that SDF format specifies the nets ending at output ports of the design are not part of the SDF file because the delay of this net is dependent on the external environment.

-max_line *num*

Reports the specified maximum number of lines for the **-list_annotated** and **-list_not_annotated** options.

-list_annotated

Lists timing arcs that are back-annotated. Use this option to identify the specified annotated delays if no annotated delays are expected. This option lists both connected and unconnected cells.

-list_not_annotated

Lists timing arcs that are not back-annotated. Use this option to identify which delay arcs are not annotated. This option lists both connected and unconnected cells.

-constant_arcs

Keeps a separate count for the arcs that are disabled due to logic constants (but not case_analysis). With this option, the **-list_annotated** or **-list_not_annotated** option does not list the arc disabled due to logic constants. The total count at the end of the table includes the number of constant arcs.

DESCRIPTION

Provides a summary report of how many cell and net delays are annotated in the current design. The purpose of this command is to check if all delays of the design are annotated after reading an SDF file. By default, net delay starting from input ports or ending at output ports are considered separately.

By default, this command reports annotated data on both cells and nets. To report annotated data on cells or nets only, use the **-cell** or **-net** option, respectively.

EXAMPLES

The following is an example of a default report showing annotated delays:

```
pt_shell> report_annotated_delay

******************************
report: annotated_delay
******************************

                             |           |           |    NOT    |
                             |   Total   | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell arcs                    |        19 |        19 |         0 |
cell arcs (unconnected)      |         1 |         1 |         0 |
internal net arcs            |         3 |         3 |         0 |
net arcs from primary inputs |        11 |        11 |         0 |
net arcs to primary outputs  |         4 |         4 |         0 |
-----------------------------+-----------+-----------+-----------+
                             |        38 |        38 |         0 |
```

The following is an example of a delay annotation report that includes a separate count for constant arcs.

```
pt_shell> report_annotated_delay -constant_arcs

******************************
report: annotated_delay
******************************

                             |           |           |    NOT    |
                             |   Total   | Annotated | Annotated |
```

```
-----------------------------+-----------+-----------+-----------+
cell arcs                    |        19 |        17 |         0 |
cell arcs (unconnected)      |         1 |         1 |         0 |
constant arcs                |           |         2 |         0 |
internal net arcs            |         3 |         3 |         0 |
constant arcs                |           |         0 |         0 |
net arcs from primary inputs |        11 |        11 |         0 |
constant arcs                |           |         0 |         0 |
net arcs to primary outputs  |         4 |         4 |         0 |
constant arcs                |           |         0 |         0 |
-----------------------------+-----------+-----------+-----------+
                             |        38 |        38 |         0 |
```

SEE ALSO

read_sdf(2)
remove_annotated_delay(2)
reset_design(2)
set_annotated_delay(2)

report_annotated_mtcmos_switch_cell

Reports MTCMOS switch cells and their related annotations in the design.

SYNTAX

```
status report_annotated_mtcmos_switch_cell
   object_list
   -on_state_resistance | -off_state_leakage | -state
   -switch_pin_group
   -verbose
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

Specifies a list of MTCMOS switch cell instance objects to be reported. Items within the list can be name patterns or collections of MTCMOS switch cell instance objects.

-on_state_resistance

Indicates that on-state resistance is to be reported.

-off_state_leakage

Indicates that off-state leakage is to be reported.

-state

Indicates that the on/off state is to be reported.

-switch_pin_group

Reports annotations for the switch cells that are of the same switch cell group. When used with the -verbose option, the command displays netlist connectivity information for switch cells which are connected in the daisy chain, including input pins, output pins, and connecting cells.

-verbose

Indicates that the report is to show verbose MTCMOS switch cell information, including library, connected supply networks, control pins, and control networks.

DESCRIPTION

The command reports the annotations on the MTCMOS switch cells in the design.

The default information reported for a MTCMOS switch cell includes its reference cell name, library when condition, input PG pin, output PG pin, user-annotated value, and library value. See the **set_annotated_mtcmos_switch_cell** man page for information about how to specify on-state resistance, off-state leakage and on/off state.

The -verbose option reports extra information, such as library, supply networks, control pins, and control networks.

EXAMPLES

The following example reports on-state resistance for all MTCMOS switch cells in the design, including both library and user-annotated information:

```
pr_shell> report_mtcmos_switch_cell -on_state_resistance [get_cell -filter is_mtcmos_switch_cell]
****************************************
Report : report_mtcmos_switch_cell
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************

                            Input  Output Annotated   Library      Current
Cell     Ref    Pin State   PG Pin PG Pin Resistance  Resistance   Pin State
-----------------------------------------------------------------------------
PM1      PM     sw1 1 sw2 0 VDD    VVDD   1000.0      2000.0       No
PM1      PM     sw1 0 sw2 1 VDD    VVDD   100.0       300.0        Yes
PM1      PM     sw1 1 sw2 1 VDD    VVDD   90.0        100.0        No
```

The following example generates a verbose report with detailed information, including the switch cell instance's reference name and the library from where it is contained:

```
pr_shell> report_mtcmos_switch_cell -verbose [get_cell -filter is_mtcmos_switch_cell]
****************************************
Report : report_mtcmos_switch_cell
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************

MTCMOS switch cell: pm1
-------------------------------------------------------------
  Reference: PM1
  Library: stdlibrary

  Current Pin State : No
  Pin state name: high
  Pin state : sw1 1 sw2 0
  Annotated on_state_resistance : 1000.0
  Library effective on_state_resistance : 2000.0

    Input PG pin: VDD {VDD}
    Output PG pin: VVDD {VDD_I1}
    When_str: sw1
```

```
    Control pin: sw1 {N11} sw2 {N12}
    Library channel on_state_resistance : 2000.0

  Current Pin State : Yes
  Pin state name : mid
  Pin state : sw1 0 sw2 1
  Annotated on_state_resistance : 100.0
  Library effective on_state_resistance : 300.0

    Input PG pin: VDD {VDD}
    Output PG pin: VVDD {VDD_I1}
    When_str: sw2
    Control pin: sw1 {N11} sw2 {N12}
    Library channel on_state_resistance : 300.0

  Current Pin State : No
  Pin state name : low
  Pin state : sw1 1 sw2 1
  Annotated on_state_resistance : 90.0
  Library effective on_state_resistance : 100.0 (from set_mtcmos_switch_lib_cell)


    Input PG pin: VDD {VDD}
    Output PG pin: VVDD {VDD_I1}
    When_str: sw1
    Control pin: sw1 {N11} sw2 {N12}
    Library channel on_state_resistance : 2000.0

    Input PG pin: VDD {VDD}
    Output PG pin: VVDD {VDD_I1}
    When_str: sw2
    Control pin: sw1 {N11} sw2 {N12}
    Library channel on_state_resistance : 300.0
```

SEE ALSO

get_cell(2)
set_annotated_mtcmos_switch_cell(2)
set_mtcmos_switch_lib_cell(2)

report_annotated_parasitics

Reports net parasitics back-annotated on the current design.

SYNTAX

```
string report_annotated_parasitics
   [-check]
   [-no_check]
   [-internal_nets]
   [-boundary_nets]
   [-driverless_nets]
   [-loadless_nets]
   [-pin_to_pin_nets]
   [-max_nets num]
   [-list_annotated]
   [-list_not_annotated]
   [-list_lumped]
   [-constant_arcs]
   [net_list]
```

Data Types

```
num         integer
net_list    list
```

ARGUMENTS

-check

Checks the design to verify that all annotated RC networks are complete. This option causes the **report_annotated_parasitics** command to check that all fanouts of each net connect through the RC network to each driver of the net. An error message reports nets with incomplete RC networks. This option is the default for PrimeTime version C-2009.06 and later.

-no_check

Prevents checking of the design to verify that all annotated RC networks are complete.

-internal_nets

Reports only parasitics annotated on internal nets (nets connected only to cell pins).

-boundary_nets

only parasitics annotated on boundary (port) nets. A connection to any port qualifies a net as a boundary net.

-driverless_nets

Reports only parasitics annotated on driverless nets (nets that do not have a global driver).

-loadless_nets

Reports only parasitics annotated on loadless nets (nets that do not have a global load) are reported. Note that if there are nets that neither have a global driver nor a global load, those nets are counted as driverless nets and not as loadless nets for the report.

-pin_to_pin_nets

Reports only parasitics annotated on nets that have at least one global driver and at least one global load pin.

-max_nets *num*

Limites reporting to the specified maximum number of nets for the **-list_annotated**, **-list_not_annotated** and **-list_lumped** options.

-list_annotated

Lists back-annotated nets. You cannot use this option together with the **-list_not_annotated** or **-list_lumped** options.

-list_not_annotated

Lists nets that are not back-annotated. You cannot use this option together with the **-list_annotated** or **-list_lumped** options.

-list_lumped

Lists lumped nets. You cannot use this option together with the **-list_annotated** or **-list_not_annotated** options.

-constant_arcs

Keeps a separate count for nets connected to pins with arcs that are disabled due to logic constants or case analysis. With this option, the **-list_annotated** or **-list_not_annotated** option does not list the constant nets. The total count at the end of the table includes the count of constant nets, and a count of constant nets suppressed (due to **-max_nets**) is shown separately from the count of ordinary nets suppressed.

net_list

Reports only the specified nets.

DESCRIPTION

This command reports nets annotated with parasitics in the current design. This command can also check the consistency of parasitics after reading ascii or binary parasitics. The report summarizes how many nets are annotated with reduced parasitics (pi models) or detailed parasitics (RC networks).

Parasitics are associate with global nets. This report counts nets without considering hierarchical crossings - only one segment per global net is reported.

The report clearly shows how many nets are back-annotated with lumped, pi and detailed RC networks. The nets that are not annotated are shown in a separate column. In addition, nets that do not have a global driver and nets that do not have any global load pins are shown separately. If a net falls into both these categories, that is if a net does not have a driver and does not have a load, it is shown in driver-less nets row.

Details on which nets are annotated or not annotated can be displayed using either the **-list_annotated** or **-list_not_annotated** option. Listing annotated nets show a detailed description of RC networks. Listing unannotated nets show only a list of net names. By default, these reports only show 10 nets. To increase that limit, use the **-max_nets** option.

You can also report parasitics for a specific set of nets by using the *net_list* option. The **-list_annotated** and **-list_not_annotated** options restrict their scope to the nets in *net_list*.

When the **si_enable_analysis** variable is set, a column for coupled networks is added to the summary report that provides the number nets that have coupling capacitors.

If the **-check** option is used, and the command reports nets with incomplete parasitics, you can complete partially annotated nets using the **complete_net_parasitics** command.

EXAMPLES

The following is an example of an annotated parasitics report.

```
pt_shell> report_annotated_parasitics -check

******************************
report: annotated_parasitics
        -check
        -internal_nets
        -boundary_nets
******************************

                     |         |         |         |   RC    |   Not   |
Net Type             |  Total  | Lumped  |  RC pi  | network |Annotated|
---------------------+---------+---------+---------+---------+---------+
Internal nets        |         |         |         |         |         |
  - Pin to pin nets  |   23645 |       0 |       0 |   23644 |       1 |
  - Driverless nets  |       3 |       0 |       0 |       0 |       3 |
  - Loadless nets    |       1 |       0 |       0 |       0 |       1 |
---------------------+---------+---------+---------+---------+---------+
Boundary/port nets   |         |         |         |         |         |
  - Pin to pin nets  |      34 |       0 |       0 |      34 |       0 |
  - Driverless nets  |       0 |       0 |       0 |       0 |       0 |
  - Loadless nets    |       0 |       0 |       0 |       0 |       0 |
---------------------+---------+---------+---------+---------+---------+
                     |   23683 |       0 |       0 |   23678 |       5 |
```

The following is an example of an annotated parasitics report for a coupled network.

```
pt_shell> report_annotated_parasitics -check

******************************
report: annotated_parasitics
        -check
        -internal_nets
        -boundary_nets
******************************

                     |         |         |         |   RC    | Coupled |   Not   |
Net Type             |  Total  | Lumped  |  RC pi  | network | network |Annotated|
---------------------+---------+---------+---------+---------+---------+---------+
Internal nets        |         |         |         |         |         |         |
  - Pin to pin nets  |   23645 |       0 |       0 |    5453 |   18191 |       1 |
```

```
  - Driverless nets |        3 |        0 |        0 |        0 |        0 |        3 |
  - Loadless nets   |        1 |        0 |        0 |        0 |        0 |        1 |
--------------------+----------+----------+----------+----------+----------+----------+
Boundary/port nets  |          |          |          |          |          |          |
  - Pin to pin nets |       34 |        0 |        0 |       12 |       22 |        0 |
  - Driverless nets |        0 |        0 |        0 |        0 |        0 |        0 |
  - Loadless nets   |        0 |        0 |        0 |        0 |        0 |        0 |
--------------------+----------+----------+----------+----------+----------+----------+
                    |    23683 |        0 |        0 |     5465 |    18213 |        5 |
```

The following is an example of an annotated parasitics report, with PrimeTime SI analysis off, showing constant nets.

```
pt_shell> report_annotated_parasitics -check -constant_arcs

******************************
report: annotated_parasitics
        -check
        -internal_nets
        -boundary_nets
        -constant_arcs
******************************

                    |          |          |          |   RC     | Coupled |   Not    |
Net Type            |  Total   | Lumped   |  RC pi   | network  | network |Annotated|
--------------------+----------+----------+----------+----------+----------+----------+
Internal nets       |          |          |          |          |          |          |
  - Pin to pin nets |    23640 |        0 |        0 |     5453 |    18186 |        1 |
  - Driverless nets |        3 |        0 |        0 |        0 |        0 |        3 |
  - Loadless nets   |        1 |        0 |        0 |        0 |        0 |        1 |
  - Constant nets   |        5 |        0 |        0 |        0 |        5 |        0 |
--------------------+----------+----------+----------+----------+----------+----------+
Boundary/port nets  |          |          |          |          |          |          |
  - Pin to pin nets |        6 |        0 |        0 |       12 |       22 |        0 |
  - Driverless nets |        0 |        0 |        0 |        0 |        0 |        0 |
  - Loadless nets   |        0 |        0 |        0 |        0 |        0 |        0 |
  - Constant nets   |        0 |        0 |        0 |        0 |        0 |        0 |
--------------------+----------+----------+----------+----------+----------+----------+
                    |    23683 |        0 |        0 |     5465 |    18213 |        5 |
```

SEE ALSO

complete_net_parasitics(2)
read_parasitics(2)
remove_annotated_parasitics(2)
reset_design(2)
si_enable_analysis(3)

report_annotated_power

Reports annotated power.

SYNTAX

```
status report_annotated_power
   [-list_annotated]
   [-rails rail_list]
```

Data Types

rail_list	list

ARGUMENTS

-list_annotated

Reports a list of cells with power values annotated on them.

-rails *rail_list*

Reports the annotated power for the specified power supply nets (or rails in non-UPF mode). This feature is only valid if the **power_enable_multi_rail_analysis** variable is set to **true**.

DESCRIPTION

This command provides a summary report of annotated powers in the current design.

EXAMPLES

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -internal_power 1 -leakage_power 0.01 u0/*
1
pt_shell> report_annotated_power -list_annotated

****************************************
Report : annotated_power
        -list_annotated
****************************************
```

```
Annotated cell powers:
------------------------------


 1.  u0/u0  (internal: 1  leakage: 0.01)

 2.  u0/u1  (internal: 1  leakage: 0.01)



                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell     |        2  |        1  |        1  |
leaf cell                     |        3  |        1  |        2  |
-----------------------------+-----------+-----------+-----------+
                              |        5  |        2  |        3  |

1
pt_shell> remove_annotated_power u0/u0
1
pt_shell> report_annotated_power

****************************************
Report : annotated_power
****************************************

                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell     |        2  |        1  |        1  |
leaf cell                     |        3  |        0  |        3  |
-----------------------------+-----------+-----------+-----------+
                              |        5  |        1  |        4  |

1
```

SEE ALSO

```
set_annotated_power(2)
remove_annotated_power(2)
```

report_app_var

Shows the application variables.

SYNTAX

```
string report_app_var
   [-verbose]
   [-only_changed_vars]
   [pattern]
```

Data Types

```
pattern      string
```

ARGUMENTS

-verbose

Shows detailed information.

-only_changed_vars

Reports only changed variables.

pattern

Reports on variables matching the pattern. The default is "*".

DESCRIPTION

The **report_app_var** command prints information about application variables matching the supplied pattern. By default, all descriptive information for the variable is printed, except for the help text.

If no variables match the pattern, an error is returned. Otherwise, this command returns the empty string.

If the **-verbose** option is used, then the command also prints the help text for the variable. This text is printed after the variable name and all lines of the help text are prefixed with " #".

The Constraints column can take the following forms:

{val1 ...}

The valid values must belong to the displayed list.

val \<= a

The value must be less than or equal to "a".

val \> = b

The value must be greater than or equal to "b".

b \<= val \<= a

The value must be greater than or equal to "b", and less than or equal to "a".

EXAMPLES

The following are examples of the **report_app_var** command:

```
prompt> report_app_var sh*
Variable                 Value     Type    Default   Constraints
------------------------ --------- ------- --------- ----------------------
sh_continue_on_error     false     bool    false
sh_script_stop_severity  none      string  none      {none W E}
 \.\.\.


prompt> report_app_var sh* -verbose
Variable                 Value     Type    Default    Constraints
------------------------ --------- ------- --------- ----------------------
sh_continue_on_error     false     bool    false
   # Allows source to continue after an error
sh_script_stop_severity  none      string  none      {none W E}
   # Indicates the error message severity level which would cause
   # a script to stop executing before it completes
 \.\.\.
```

SEE ALSO

```
get_app_var(2)
set_app_var(2)
write_app_var(2)
```

report_attribute

Reports the attributes on one or more objects.

SYNTAX

```
string report_attribute
   [-class class_name]
   [-nosplit]
   [-application]
   [-attributes attribute_list]
   [-summary]
   object_list
```

Data Types

```
class_name          string
attribute_list      list
object_list         list
```

ARGUMENTS

-class *class_name*

If *object_list* is a name, this is its class. Allowable values are **design**, **port**, **cell**, **pin**, **net**, **lib**, **lib_cell**, or **lib_pin**.

-nosplit

Does not split lines if column overflows.

-application

Lists application attributes as well as user-defined attributes.

-attributes *attribute_list*

Reports only the specified set of attributes for each object in *object_list*.

-summary

Prints a summary report in a column format. You can use this option only with the **-attributes** option.

object_list

List of objects to report. Each element in the list is either a collection or a pattern that combines with the *class_name* value to find the objects.

DESCRIPTION

Generates a report of attributes on the specified objects. By default, only user-defined attributes are displayed. Using the **-application** option adds application attributes to the report. For application attributes that are of the type **collection**, the name of the first object in the collection is displayed.

To limit the reported attributes to a specific set, use the **-attributes** option.

EXAMPLES

This example defines an attribute 'X' for cells, sets the value on all cells in this level of the hierarchy, and reports the result.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'

pt_shell> report_attribute [get_cells *]

****************************************
Report : Attribute
Design : my_des
****************************************

Design     Object        Type      Attribute Name    Value
-------------------------------------------------------------
my_des     i1            int       X                 30
my_des     i2            int       X                 30
```

This example shows how application attributes can display.

```
pt_shell> report_attribute -application [get_designs my_des]

****************************************
Report : Attribute
Design : my_des
****************************************

Design   Object    Type      Attribute Name    Value
-------------------------------------------------------
my_des   my_des    string    full_name         my_des
my_des   my_des    string    object_class      design
my_des   my_des    float     area              56.000000
my_des   my_des    string    tree_type_max     balanced_case
my_des   my_des    string    tree_type_min     balanced_case
my_des   my_des    float     wire_load_min_block_size
                                               0.000000
my_des   my_des    string    wire_load_mode    top
```

In the following example, the **-attributes** option reports details about the timing arcs in a cell in the summary format.

```
pt_shell> report_attribute [get_timing_arcs -of_object [get_cell ffa]] \
          -attributes {from_pin to_pin is_disabled delay_max_rise \
           delay_min_rise} -summary
```

```
****************************************
Report : Attribute
Design : my_des
****************************************

Object     from_pin      to_pin        is_disabled   delay_max_rise delay_min_rise
--------------------------------------------------------------------------------
arc        ffa/CP        ffa/D         false         0.400000       0.400000
arc        ffa/CP        ffa/D         false         0.850000       0.850000
arc        ffa/CP        ffa/Q         false         1.335800       1.335800
arc        ffa/CP        ffa/QN        false         2.383800       2.383800
arc        ffa/CD        ffa/Q         false         0.000000       0.000000
arc        ffa/CD        ffa/QN        false         0.000000       0.000000
```

SEE ALSO

collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
remove_user_attribute(2)
set_user_attribute(2)

report_bottleneck

Reports timing bottleneck information.

SYNTAX

```
string report_bottleneck
   [-cost_type path_count | path_cost | fanout_endpoint_cost]
   [-delay_type max | min]
   [-slack_lesser_than slack_limit]
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-max_cells cell_count]
   [-max_paths path_count]
   [-nworst_paths paths_per_endpoint]
   [-group group_name]
   [-significant_digits digits]
   [-nosplit]
```

Data Types

slack_limit	float
from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
cell_count	int
path_count	int
paths_per_endpoint	int
group_name	list
digits	int

ARGUMENTS

-cost_type path_count | path_cost | fanout_endpoint_cost

Specifies a cost type to use for computing the bottleneck cost:

- **path_count** (the default) - Uses the number of violating paths through the cell.
- **path_cost** - Uses the total slack of violating paths through the cell.
- **fanout_endpoint_cost** - Uses the total cost of violating endpoints in the fanout of the cell.

-delay_type max | min

Specifies whether to compute the bottleneck cost for

- **max** (the default) - Setup analysis.
- **min** - Hold analysis.

-slack_lesser_than *slack_limit*

Considers only those paths with a slack less than *slack_limit* to find bottlenecks.

-from *from_list*

Considers only paths from the specified pins, ports, nets, or startpoints clocked by named clocks to find bottlenecks.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-to *to_list*

Considers only paths to the specified pins, ports, nets, or endpoints clocked by specified clocks.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-through *through_list*

Considers only paths through the specified pins, ports, nets or cells to find bottlenecks.

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

-rise_through ***rise_through_list***

Same as the **-through** option, except that the path must rise through the specified objects.

-fall_through ***fall_through_list***

Same as the **-through** option, except that the path must fall through the specified objects.

-max_cells ***cell_count***

Specifies the number of cells to report. The default is 20.

-max_paths ***path_count***

Specifies the number of paths to be considered per path group. Allowed values are 1 to 2000000; the default is to use the *paths_per_endpoint* setting.

-nworst_paths ***paths_per_endpoint***

Specifies the number of paths to be considered per endpoint. Allowed values are 1 to 2000000; the default is 100.

-group ***group_name***

Indicates that only paths in *group_name* are to be considered.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default is 2. Use this option if you want to override the default.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

This command reports information about timing bottlenecks in the current design. A bottleneck is a common point in the design that contributes to multiple violations.

To report bottlenecks throughout the current design, the arrival totals and slacks must be available on all pins, not just at endpoints. To achieve this, set the **timing_save_pin_arrival_and_slack** variable to **true** before using this command. If the **timing_save_pin_arrival_and_slack** variable is set to **false**, this command automatically sets it to **true** and updates the design timing before the command executes.

If you intend to use this command, it is recommended that you set the **timing_save_pin_arrival_and_slack** variable to **true** before the first timing update, therefore preventing the cost of an additional timing update.

EXAMPLES

The following example reports the 20 worst bottleneck cells in the design based on the number of paths through the cell with slack less than 0.0.

```
pt_shell> report_bottleneck

****************************************
Report : bottleneck
        -max_cells 20
        -nworst_paths 100
Design : example
 ...
****************************************

Bottleneck Cost = Number of violating paths through cell

                                                    Bottleneck
Cell                                Reference       Cost
-----------------------------------------------------------------
ipxr/ir1/i2/i0/u3                   DFF1A           39656.00
ipxr/ir1/U14                        INV2            39654.00
ipxr/ir1/U16                        INV8            39654.00
ipxr/ir1/i2/i0/u4                   DFF1A           31806.00
ipxr/ir1/U15                        BUF8B           31804.00
ipxr/U1712                          MUX2I           23999.00
ipxr/U558                           INV4            23999.00
ipxr/U1370                          NAN4            22485.00
ipxr/x01                            INV2            22485.00
dmac/BufEn                          BUF3            22485.00
dmac/U574                           INV             22485.00
ipxr/U1335                          NAN6CH          21844.00
ipxr/U564                           MUX2I           20074.00
ipxr/U1694                          MUX2A           19936.00
ipxr/U1559                          BUF2C           14785.00
ipxr/U1484                          MUX2I           14252.00
ipxr/U1486                          MUX2I           12468.00
SccMOD/U929                         OR3             12135.00
ipxr/U1493                          MUX2I           11248.00
dmac/BiuSMOD/U879                   NAN2            10949.00
```

SEE ALSO

```
report_cell(2)
report_timing(2)
report_default_significant_digits(3)
timing_save_pin_arrival_and_slack(3)
```

report_bus

Reports the bused ports or nets in the current instance or current design.

SYNTAX

```
string report_bus
   [-nosplit]
```

ARGUMENTS

-nosplit

Does not split lines if column overflows.

DESCRIPTION

Displays information about buses (pins or nets) in the current instance or current design. If the current instance is set, the report is generated for the design of that instance; otherwise, the report is generated for the current design.

EXAMPLES

The following command reports the buses in the design.

```
pt_shell> report_bus

****************************************
Report : bus
Design : new_design
****************************************

Bussed Port     Dir      From     To    Width
-------------------------------------------------------------
acnt            out      15       0     16
baddreg         out      15       0     16
bwordreg        out      15       0     16
caddreg         out      15       0     16
```

SEE ALSO

`report_port(2)`

report_case_analysis

Reports case analysis entries on ports and pins, and the propagation of user-specified case values and logic constant values throughout the netlist.

SYNTAX

```
status report_case_analysis
  [-all]
  [-nosplit]
  [-to]
  [-from]
  [-sequential_propagation]
  [pin_or_port_list]
```

Data Types

```
pin_or_port_list      list
```

ARGUMENTS

-all

Reports all pins that have set case analysis values and reports the built-in constant pins of the design that are considered for startpoints of logic constant propagation. Logic constant propagation is performed by default from the constant pins of the design, unless the **disable_case_analysis** variable is set to **true**.

-nosplit

Prevents line splitting to facilitate parsing of information from the report output.

-to

Shows the backward trace of case propagation to each pin or port specified by *pin_or_port_list*. The backward trace expands only to pins or ports with case values that contribute to the case value on the destination pin or port. If the number of pins or ports involved in backward trace exceeds the number specified by **-max_objects**, PrimeTime truncates the trace and prints a message at the end of the trace to indicate an incomplete trace.

-from

Shows the forward trace of case propagation from each pin or port specified by *pin_or_port_list*. The forward trace expands only to pins or ports where the case value from the source contributes to the case value on that pin or port. If the number of pins or ports involved in the forward trace exceeds the number specified by **-max_objects**, PrimeTime truncates the trace and prints a message at the end of the trace to indicate an incomplete trace.

-sequential_propagation

Reports sequential cell instances and library cells that are enabled for constant propagation. To enable constant propagation on specific sequential cells, set the **case_analysis_sequential_propagation** variable to **never**, and run the **set_case_sequential_propagation** command.

pin_or_port_list

Reports the propagated case values on the specified pins or ports.

DESCRIPTION

This command reports case analysis settings on ports and pins that are specified by the **set_case_analysis** command.

If you do not specify *pin_or_port_list*, the report lists all pins and ports on which case analysis is set; this report does not show where the logic constant values are propagated.

EXAMPLES

The following example specifies that pins U1/U2/A and U1/U3/CI are set to a constant logic I1:

```
pt_shell> set_case_analysis 1 {U1/U2/A U1/U3/CI}
pt_shell> report_case_analysis

****************************************
Report : case_analysis
 ...
****************************************

Pin name                     User case analysis value
-----------------------------------------------------
U1/U2/A                       1
U1/U3/CI                      1
```

The following example displays the fanout trace of the case value on port CL:

```
pt_shell> report_case_analysis -from CL

****************************************
Report : case_analysis
Design : counter
 ...
****************************************
Properties         Value   Pin/Port
----------------------------------------
user case          1       CL


Case fanout report:
Verbose Forward Trace for pin/port CL:

Pin/Port ID  Value  Fanout Pin/Port IDs  Pin/Port
--------------------------------------------------
0            1      1 8                 CL
1            1      3                   p/B
2            1      3                   p/A
```

```
3            1      4 5 6 7              p/Z
4            1                           a/D
5            1                           j/D
6            1                           g/D
7            1                           d/D
8            1                           o/A
9            0                           o/B
```

The following example enables sequential propagation on cell instances (FF1, FF2, and FF3), library cells (D_FF, D_FF_R, and D_FF_S) and then prints them using the '-sequential_propagation' option in the report_case_analysis command.

```
pt_shell>set_case_sequential_propagation { FF1 FF2 FF3 }
pt_shell>report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
 ...
****************************************

----------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 3
----------------------------------------------------------------
Cell Instance : 'FF1'
Cell Instance : 'FF2'
Cell Instance : 'FF3'
----------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 3
----------------------------------------------------------------
Lib Cell : 'snps_test/D_FF_R'
Lib Cell : 'snps_test/D_FF'
Lib Cell : 'snps_test/D_FF_S'
```

SEE ALSO

```
remove_case_analysis(2)
remove_case_sequential_propagation(2)
set_case_analysis(2)
set_case_sequential_propagation(2)
disable_case_analysis(3)
```

report_cell

Reports cell information.

SYNTAX

```
status report_cell
   [-connections]
   [-verbose]
   [-nosplit]
   [-significant_digits digits]
   [cell_list]
```

Data Types

```
digits        integer
cell_list     list
```

ARGUMENTS

-connections

Displays the pins and the nets to which they connect.

-verbose

Displays detailed connection information. For each input pin, displays the net and the driver pins on that net. For each output pin, displays the net and the load pins on that net. Use this option together with the **-connections** option.

-nosplit

Prevents line splitting in the report when a field exceeds the column width.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point shown in the report. Allowed values are 0-13. The default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option if you want to override the default. Because the tool uses fixed-precision floating-point arithmetic for delay calculation, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

cell_list

Specifies the cells to report. If you do not specify this option, all cells in the current instance are listed.

DESCRIPTION

This command displays information and statistics about cells in the current instance or current design. If you set for the current instance, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

If any cell uses nondefault operating conditions, the **report_cell** command shows information about the operating conditions for all cells.

Some information, such as whether the cell is in an ideal network or not, is displayed after a timing update (as a result of manually executing the **update_timing** command function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about cells without incurring the cost of a complete timing update.

EXAMPLES

The following example displays a report of the cells in the current design.

```
pt_shell> report_cell

****************************************
Report : cell
 ...
****************************************

Attributes:
    b - black-box (unknown)
    h - hierarchical
    n - noncombinational
    u - contains unmapped logic
    A - abstracted timing model
    E - extracted timing model
    S - Stamp timing model
    Q - Quick timing model (QTM)
    I - ideal network

Cell                       Reference       Library             Area  Attributes
--------------------------------------------------------------------------------
i1                         inter                                6.00  h
low_i                      low                                  2.00  h
n0_i                       ND2             my_lib               1.00
o_reg2                     FD2             my_lib               9.00  n
--------------------------------------------------------------------------------
Total 4 cells                                                  18.00
```

The following example gives a verbose report of the cell connections.

```
pt_shell> report_cell -verbose -connections

****************************************
Report : cell
        -connections
        -verbose
 ...
****************************************

Connections for cell 'i1':
```

```
    Reference:          inter
    Hierarchical:       TRUE
    Area:               6

    Input Pins          Net            Net Driver Pins   Driver Pin Type
    ----------------    ------------   ----------------  ----------------
    a                   out_1          o_reg1/Q          Output Pin (FD2)
    b                   out_2          o_reg2/Q          Output Pin (FD2)
    c                   out_3          o_reg3/Q          Output Pin (FD2)
    d                   out_4          o_reg4/Q          Output Pin (FD2)

    Output Pins         Net            Net Load Pins     Load Pin Type
    ----------------    ------------   ----------------  ----------------
    z1                  i1t1           low_i/n1/A        Input Pin (NR4)
    z2                  i1t2           low_i/n1/B        Input Pin (NR4)
    z3                  i1t3           low_i/n1/C        Input Pin (NR4)

Connections for cell 'low_i':
    Reference:          low
    Hierarchical:       TRUE
    Area:               2

    Input Pins          Net            Net Driver Pins   Driver Pin Type
    ----------------    ------------   ----------------  ----------------
    a                   i1t1           i1/low/n1/Z       Output Pin (NR4)
    b                   i1t2           i1/low2/n1/Z      Output Pin (NR4)
    c                   i1t3           i1/low3/n1/Z      Output Pin (NR4)
    d                   in2            in2               Input Port

    Output Pins         Net            Net Load Pins     Load Pin Type
    ----------------    ------------   ----------------  ----------------
    z                   low            o_reg2/D          Input Pin (FD2)

Connections for cell 'n0_i':
    Reference:          ND2
    Library:            my_lib
    Area:               1

    Input Pins          Net            Net Driver Pins   Driver Pin Type
    ----------------    ------------   ----------------  ----------------
    A                   p_in           p_in              Input Port
    B                   p_in           p_in              Input Port

    Output Pins         Net            Net Load Pins     Load Pin Type
    ----------------    ------------   ----------------  ----------------
    Z                   n0             n1_i/B            Input Pin (ND2)

Connections for cell 'o_reg2':
    Reference:          FD2
    Library:            my_lib
    Area:               9

    Input Pins          Net            Net Driver Pins   Driver Pin Type
    ----------------    ------------   ----------------  ----------------
    D                   low            low_i/n1/Z        Output Pin (NR4)
    CP                  CLOCK          CLOCK             Input Port
    CD                  OPER           OPER              Input Port

    Output Pins         Net            Net Load Pins     Load Pin Type
    ----------------    ------------   ----------------  ----------------
    Q                   out_2          i1/low3/n1/B      Input Pin (NR4)
                                       out_2             Output Port
                                       i1/low/n1/B       Input Pin (NR4)
                                       i1/low2/n1/B      Input Pin (NR4)
```

```
    QN                  NET2QNX        i2/low3/n1/B      Input Pin (NR4)
                                       i2/low/n1/B       Input Pin (NR4)
                                       i2/low2/n1/B      Input Pin (NR4)
```

In the following example, cell U2 uses nondefault operating conditions. Therefore, the **report_cell** command shows information about the operating conditions for all cells.

```
pt_shell> report_cell "B/C/u1  B/C/D/E/u2 B/C/D/E/u3 "

****************************************
Report : cell
 ...
****************************************

Attributes:
    b - black-box (unknown)
    h - hierarchical
    n - noncombinational
    u - contains unmapped logic
    A - abstracted timing model
    E - extracted timing model
    S - Stamp timing model
    Q - Quick timing model (QTM)
    I - ideal network

Cell          Reference       Library          Area  Attributes
-------------------------------------------------------------------------
u1            BUFFD1          tech90           3.23
u2            BUFFD1          tech90           3.23
u3            BUFFD1          tech90           3.23
-------------------------------------------------------------------------
Total 3 cells                                  9.68

Cell specific operating conditions and rail voltages:

         Min             Max             Min      Max
Cell     OC              OC              Voltage  Voltage
-----------------------------------------------------------------
u1       tech90/WCCOM    tech90/WCCOM
u2       tech90/new_oc   tech90/new_oc
u3       tech90/WCCOM    tech90/WCCOM
```

SEE ALSO

current_design(2)
current_instance(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_port(2)
report_reference(2)

report_cell_usage

Reports the usage of cells in a design.

SYNTAX

```
string report_cell_usage
   [-pattern_priority pattern_list]
   [-attribute attribute_name]
   [-alternative_lib_cells]
   [-show_others]
   [-power_attribute power_attribute_name]
```

Data Types

```
pattern_list            list
attribute_name          string
power_attribute_name    string
```

ARGUMENTS

-pattern_priority *pattern_list*

Specifies a list of library cell patterns starting from the highest to lowest priority.

-attribute *attribute_name*

Specifies an attribute name that can be used either to match patterns specified by the **-pattern_priority** option or to specify power attribute of each library cell. When you specify this option together with the **-pattern_priority** option, PrimeTime uses the value of the attribute instead of library cell names for pattern matching. When you specify this option without the **-pattern_priority** option, PrimeTime uses the value of the attribute as power attribute of each library cell.

-alternative_lib_cells

Shows alternative library cells. This option will be obsolete in a future release; use the **report_eco_library_cells** command instead.

-show_others

Shows library cells that do not match the specified pattern. This option will be obsolete in a future release; use the **report_eco_library_cells** command instead.

-power_attribute *power_attribute_name*

Specifies a power attribute set on library cells. If you specify this option, the summary of power attribute value for each cell group is reported in a separate column.

DESCRIPTION

This command reports the summary of cell usage in a design. This command must be used by slaves for multi-scenario analysis flow within PrimeTime.

This command reports the usage of cells based on the specified pattern when the **-pattern_priority** option is used. If the **-pattern_priority** option is not used, the command reports summary of cell usage for each cell group. There are four cell groups:

- Combinational - Combinational cells.
- Sequential - Sequential cells such as latches and flip-flops.
- Clock - Cells in the clock network.
- Others - Other cells including io_pad, memory, and block_boxes.

EXAMPLES

The following example reports HVT and LVT cell counts in the design:

```
pt_shell> report_cell_usage -pattern_priority {HVT LVT}

****************************************
Report : cell_usage
        -pattern_priority { HVT LVT }
Design : design
 ...
****************************************

Pattern   Combinational cell     Sequential cell               Total
-----------------------------------------------------------------------
HVT           1515979 ( 32%)         1291 (  0%)      1517270 ( 32%)
LVT           2474715 ( 53%)       696222 ( 15%)      3170937 ( 68%)
Others           4115 (  0%)          532 (  0%)         4647 (  0%)
```

The following example reports summary of cell count and area for each cell group:

```
pt_shell> report_cell_usage

****************************************
Report : cell_usage
Design : design
****************************************

Cell Group                   Count                   Area
-------------------------------------------------------------
Combinational             5 ( 71%)            33.16 ( 59%)
Sequential                1 ( 14%)            16.23 ( 29%)
Clock                     1 ( 14%)             6.35 ( 11%)
Others                    0 (  0%)             0.00 (  0%)
-------------------------------------------------------------
Total                     7 (100%)            55.74 (100%)
```

The following example reports summary of cell count, area, and power attribute for each cell group:

```
pt_shell> report_cell_usage -power_attribute pwr_attr

****************************************
Report : cell_usage
        -power_attribute pwr_attr
Design : design
****************************************

Cell Group            Count                    Area             Power_Attr
--------------------------------------------------------------------------------
Combinational             5 ( 71%)           33.16 ( 59%)         1815.87 ( 66%)
Sequential                1 ( 14%)           16.23 ( 29%)          548.05 ( 20%)
Clock                     1 ( 14%)            6.35 ( 11%)          390.13 ( 14%)
Others                    0 (  0%)            0.00 (  0%)            0.00 (  0%)
--------------------------------------------------------------------------------
Total                     7 (100%)           55.74 (100%)         2754.05 (100%)
```

SEE ALSO

fix_eco_leakage(2)
fix_eco_power(2)

report_clock

Reports clock-related information.

SYNTAX

```
status report_clock
   [-attributes]
   [-skew]
   [-groups]
   [-map]
   [-map_of instance_list]
   [-cells hierarchical_cell_list]
   [-include internal | virtual]
   [-exclusivity]
   [-nosplit]
   [clock_names]
```

Data Types

```
instance_list              list
hierarchical_cell_list     list
clock_names                list
```

ARGUMENTS

-attributes

Shows clock attributes and provides a list of all the clocks in the current design. The information for each clock includes source type, signal rise and fall times, and attributes. This report is shown by default.

-skew

Reports clock latency (source and network latency) and uncertainty information set on the design by the **set_clock_latency** and **set_clock_uncertainty** commands, respectively.

Clock network latency information includes rise latency and fall latency. Clock source latency information includes rise latency and fall latency for early and late arrivals. Clock uncertainty information includes intraclock setup or hold uncertainty and interclock setup or hold uncertainty. This option also reports any fixed clock transition set by using the **set_clock_transition** command. This option only reports active clocks.

-groups

Shows the current setting of clock groups, including the list of active clocks in the current analysis scope and grouping of exclusive clocks and asynchronous clocks set by using the **set_clock_groups** command.

-map

Shows the map between top level clock and block level clock in HyperScale.

-map_of ***instance_list***

Shows the map between top level clock and block level clock in these specified HyperScale instances. If **-map_of** is not used, maps of all HyperScale instances are printed with **-map**.

-cells ***hierarchical_cell_list***

Shows clocks that are physically entering the hierarchical cell through certain hierarchical pins of the cells.

-include internal | virtual

Used together with **-cells** option, to also optionally show the internal clocks of the hierarchical cell and virtual clocks that are not physically entering the cell but launching/capturing data into/from the cell. Valid values are **internal** and **virtual**. If **-include internal** is used, the Pins column shows the source pins of the internal clock. If **-include virtual** is used, the Pins column shows the relevant data pins of the virtual clock; the 'Clock type' column shows 'v' to indicate virtual clocks.

-nosplit

Specifies not to split lines if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-exclusivity

Reports the clock exclusivity points in the design, whether inferred automatically from clocks passing through MUX blocks (when the **timing_enable_auto_mux_clock_exclusivity** variable is true) or by usage of the **set_clock_exclusivity** command.

clock_names

Lists the clocks that must be reported. Substitute the list you want for *clock_names*.

DESCRIPTION

Reports clock-related information. Displays all clock-related information on a design. The report contents are controlled by the options used. If you do not specify any options, the command displays the attributes report.

When a clock is created, it defaults to being active and is added to the active clocks list. The new clock is also added to the other clocks list if some clocks are defined to be exclusive or asynchronous with any other clocks in the design. When a clock is removed, it is removed from its related lists. To see if a clock is active or not, use the **report_clock** command with the **-attribute** option.

If a dynamic component of clock latency has been specified by the **set_clock_latency** command, the **-skew** option can be used to report what values have been set. In this case, all components of clock latency will be.

Some information, such as the waveform of a generated clock or propagated skew, is displayed after a timing update (as a result of manually executing an update_timing function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to define all clocks without incurring the cost of a complete timing update.

The report generated by using the **report_transitive_fanout** command with the **-clock_tree** option shows the fanout network of every clock source in the design.

To show the clock mapping used in HyperScale between top and block, use the **report_clock -map** command. A user-specified mapping from **set_clock_map** is annotated with an asterisk (*) in the report.

To obtain a list of all clocks in the design, use the **all_clocks** command.

EXAMPLES

Here are some examples of the **report_clock** command:

```
pt_shell> create_clock -period 20.000000 [get_ports {CLK}]
pt_shell> report_clock

****************************************
Report : clock
Design : counter
 ...
****************************************

Attributes:
    p - propagated_clock
    G - Generated clock

Clock          Period   Waveform            Attrs     Sources
--------------------------------------------------------------------------
CLK             20.00   {0 10}                        {CLK}

pt_shell> set_clock_latency 2.4 [get_clocks CLK]
pt_shell> set_clock_latency 0.17 -source -early [get_clocks CLK]
pt_shell> set_clock_latency 0.19 -source -late  [get_clocks CLK]
pt_shell> set_clock_uncertainty 1.3 [get_clocks CLK]
pt_shell> set_clock_transition  1.7 [get_clocks CLK]
pt_shell> report_clock -skew

****************************************
Report : clock_skew
Design : counter
 ...
****************************************

                 Rise           Fall           Hold           Setup
Object          Delay          Delay        Uncertainty    Uncertainty
--------------------------------------------------------------------------
CLK              2.40           2.40           1.3           1.3

--------------------------------------------------------------------------
                                Source Latency
--------------------------------------------------------------------------
Object        Min Rise       Min Fall       Max Rise       Max Fall
--------------------------------------------------------------------------
CLK              0.17           0.17           0.19           0.19

                 Rise           Fall
Object          Transition     Transition
--------------------------------------------------------------------------
CLK              1.7            1.7

pt_shell> set_clock_latency -late -source -dynamic 0.5 4.5 [get_clocks clk]
pt_shell> set_clock_latency -early -source 2.5 -dynamic 0.5 [get_clocks clk]
pt_shell> report_clock -skew

****************************************
Report : clock_skew
Design : latency
```

```
 ...
****************************************

                Min Condition Source Latency      Max Condition Source Latency
-------------------------------------------------------------------------------------------
Object        Early_r Early_f  Late_r  Late_f   Early_r Early_f  Late_r  Late_f  Rel_clk
-------------------------------------------------------------------------------------------
clk (static)     2.00    2.00    4.00    4.00      2.00    2.00    4.00    4.00
                                                                                  --
clk (dynamic)    0.50    0.50    0.50    0.50      0.50    0.50    0.50    0.50
                                                                                  --
                ---------------------------------------------------------------------------
clk (total)      2.50    2.50    4.50    4.50      2.50    2.50    4.50    4.50

pt_shell> set_clock_groups -ex -group {clk1 clk3}
pt_shell> set_clock_groups -asyn -name a1 -group clk2 -group clk2
pt_shell> set_active_clocks {clk1 clk2}
pt_shell> report_clock -groups

****************************************
Report : clock_groups
 ...
****************************************

Active clocks:
  clk1  clk2

Total exclusive groups: 1.
NAME : clk1_others
         -group {clk1 clk3 }
         -group {clk2 clk4 }

Total asynchronous groups: 1.
NAME : a1
         -group {clk2 }
         -group {clk4 }

pt_shell> report_clock -map

****************************************
Report : clock
Design : BLOCK_C
 ...
****************************************

Instance              Top level clock     Block level clock
-------------------------------------------------------------
design                SYSCLK1             BLK_SYSCLK

pt_shell> report_clock -cell [get_cell {core1 core2}] -include {virtual internal}

****************************************
Report : clock of hierarchical cell
Design : wrapper
 ...
****************************************

Clock type:
    v - Virtual clock

Cell                    Clock               Clock type     Pins
-------------------------------------------------------------------------------
core1                   A                                  core1/clk1
                        B                                  core1/clk1
                        X                   v              core1/out2
```

```
                        n1_t4_CLK          v              core1/clk2
                        n_t2c4_CLK         v              core1/in
                        C                  v              core1/out1
core2                   X                                 core2/clk1
                        n1_t4_CLK          v              core2/clk2
                        n_t2c4_CLK         v              core2/out1 core2/out2
                        C                  v              core2/in

pt_shell> report_clock -exclusivity

****************************************
Report : clock
 ...
****************************************

clock exclusivity points:

    -type       mux
    -output     MUX/Y
```

SEE ALSO

all_clocks(2)
create_clock(2)
remove_clock_groups(2)
report_transitive_fanout(2)
set_active_clocks(2)
set_clock_groups(2)
set_clock_latency(2)
set_clock_map(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_clock_exclusivity(2)
timing_enable_auto_mux_clock_exclusivity(3)

report_clock_gate_savings

Reports toggle savings on clock gates.

SYNTAX

```
string report_clock_gate_savings
   [-by_clock_gate]
   [-sequential]
   [-hierarchical]
   [-nosplit]
   [-clocks clock_list]
   [-sort_by sort_method]
   [object_list]
```

Data Types

```
clock_list      list
sort_method     string
object_list     list
```

ARGUMENTS

-by_clock_gate

With this option two tables are generated. One is a summary table with column of clock name, total registers, stages, number of clock gate per stage, number of gated cells (register + clock gates) and Clock Gating Efficiency of that stage. The other is detailed clock gating structure table which contains clock name, clock gate stage, gating element, fanout, number of registers, number of gate cells, Toggle Savings, Clock Gating Efficiency, List of gate cells.

-sequential

Specifies that the report is to report on registers in the design. When the **-sequential** option is applied, the command uses toggle rates to determine the fraction of clock events at registers that result in a toggle on the Q pin. This fraction is a measure of how well clock gated a register is. Values near 25% are near optimal for datapath registers. Lower values indicate there might be more room for clock gating, as the clock is toggling often when the Q pin toggles little.

In addition to the Q/Clk toggle ratio, the Register Gating Efficiency is reported for each register. This measures the ratio between the toggle rate of the local (gated) clock, and the root clock specified with create_clock. High values indicate that most of the toggles from the root clock have been suppressed by clock gates along the clock tree from the root to the register.

When the **-by_clock_gate** and **-sequential** options are used together, the command reports on register banks in the design. Registers are grouped into clusters by the clock gates that drive their clock pins. An additional cluster of ungated registers is also reported included in the report.

-hierarchical

Specifies that results are to be aggregated into hierarchical blocks, and each block is to be reported. Without this option, individual clock gates or registers are reported. The data is aggregated by averaging toggle savings over clock gates in a module.

The **-hierarchical** option cannot be used when both the **-by_clock_gate** and **-sequential** options are also used together.

-nosplit

Specifies that report lines should not be split when names are longer than the space provided in the report. This option can make it easier for scripts to parse the results, but might make it harder for humans to read the columns.

-clocks *clock_list*

Specifies that objects not in the clock domain of one of the given clocks should be excluded from the report.

-sort_by *sort_method*

Specifies the order in which to print design objects. You can specify the **-sort** option with the **-sequential** and **-by_clock_gate** options but not with the **-hierarchical** option. You can specify the following sort methods:

```
-by_clock_gate              -sequential
----------------            --------------
name                        name
clock_gating_stage          clock_gating_stage
clock_toggle_rate           clock_toggle_rate
toggle_savings              q_clk_ratio
clock_gating_efficiency     q_toggle_rate
                            gating_efficiency
```

object_list

Specifies that the given cells are to be reported on only. If cells in the list are hierarchical, all clock gates (or registers, if the **-sequential** option is used) under the hierarchical cells are reported on. If the **-sequential** option is used without the **-by_clock_gate** option, any leaf cells in the list should be registers. If the **-by_clock_gate** option is used, or if both the **-by_clock_gate** and **-sequential** options are used, any leaf cells should be individual clock gates.

In the default report, the *object_list* option can also contain clocks, in which case only those clocks are reported on.

DESCRIPTION

This command uses toggle rates in the design to report on the toggle savings on the clock tree from clock gating in the design. To use the command, you should first apply switching activities before the command. In the time based mode, this can be done using the **read_vcd** command followed by the **update_power** command.

In the averaged power mode, apply switching activities by using the **read_saif**, **set_switching_activity**, or **read_vcd** command. In the averaged mode, partial switching activity annotation causes the tool to propagate switching activities.

The usefulness of the **report_clock_gate_savings** command is largely dependent on the quality of the switching activities. To be useful, the activities should come from simulation. The testbench used from simulation should be designed to simulate the typical behavior of the design and should cover all modes that are part of the

typical behavior. If a mode is left off, some part of the design might not be exercised.

The default report displays overview information about each clock and the registers that the clock drives. You can use this report to summarize the overall effectiveness of clock gating in the design. The report computes the average toggle savings over the registers driven by each clock, where here the toggle savings for a register is defined as the ratio of the toggle rate at the clock pin of the register to the toggle rate of the clock. A 20% toggle savings means that the toggle rate at clock pin of a register is 80% of the toggle rate of the main clock for the clock domain. When a register is in several clock domains, the register is only included with the domain of the faster clock, so that registers are not included twice in the report.

The **-sequential** reports helps to identify registers with many clk events but few Q toggles.

When the Q/Clk ratio for registers is significantly lower than 25%, there can be room to improve the clock gating for these registers by adding clock gating or increasing the fraction of time that clock gating suppresses clock toggles. A brief explanation of why 25% can be regarded as optimal is below.

In the case of a datapath, during cycles when the datapath is exercised, the clock gating is enabled, and the probability of Q toggling in a cycle is approximately 50%, assuming random and independent data is moving through the datapath. On cycles when the datapath is not being exercised, the clock gating should be disabled. In this idealized situation, the Q/clk toggle ratio should be 25%, since the clk toggles twice in a cycle, and the Q toggles on average one time every other cycle (a 1:4 ratio).

There are some exceptions where registers can have Q/Clk ratios larger than 25%, such as low bits of counters where the Q should toggle every cycle when the counter is active. However, on the whole, most registers in the design should report (using the **-sequential** option) a Q/Clk ratio less than 25%.

More information can be gathered about individual clock gates and their associated registers when the **-by_clock_gate** and **-sequential** options are used together. In this case, Individual clock gates are reported, with each clock gate followed by a list of the registers driven by the clock gate. This report can be more useful than the **-sequential** report alone in helping find register banks with low Q/Clk ratios, where it is possible to disable the clock gate more often. The report can also help find register banks that should be split; it might be that a portion of the registers driven by a clock gate have low Q/Clk ratios, and these registers might be able to have their own separate clock gate, which could allow disabling the clock more often for those registers.

The **-hierarchical** reports (which can be used with either the **-by_clock_gate** or **-sequential** option) can help to identify modules of interest. Since many register banks are too small for clock gating, the hierarchical sequential report might be less useful as the register banks of interest are overwhelmed in the aggregated data by the small register banks which are not clock gated.

USAGE

You can use the default report to view the overall effectiveness of clock gating in the design.

EXAMPLES

The following example shows the default report:

```
report_clock_gate_savings
****************************************
Report : Clock Gate Savings
         power_mode: Averaged
Design : top
****************************************

--------------------------------------------------------------------------------
Clock: clk
```

```
  + Clock Toggle Rate: 0.2
  + Number of Registers: 200
  + Number of Clock Gates: 29
  + Average Register Gating Efficiency (savings with respect to root clock): 48.1%
-----------------------------------------------------------------------------------------
Toggle Savings                        Number of       % of
Distribution                          Registers       Registers
-----------------------------------------------------------------------------------------
100%                                  8               4.0%
80% - 100%                            13              6.5%
60% - 80%                             71              35.5%
40% - 60%                             71              35.5%
20% - 40%                             11              5.5%
0%                                    24              12.0%
-----------------------------------------------------------------------------------------
```

The following example shows the **-by_clock_gate** report on a particular module. There are two individual clock gates, with toggle rates shown.

```
pt_shell> report_clock_gate_savings -by_clock_gate

****************************************
Report : Clock Gate Savings
        power_mode: Averaged
        -by_clock_gate
 ...
****************************************

-------------------------------------------------------------------------------------
                         Clock Gate Structure Summary
-------------------------------------------------------------------------------------
Clock     Total         Clock Gate     # of Clock     # of           Clock Gating
          Registers     Stage          Gates          Registers      Efficiency
-------------------------------------------------------------------------------------
CLK       12            0              0              0              NA
                        1              2              8              30.61%
                        2              1              4              54.52%
-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------
                                      Clock Gate Structure Details
-------------------------------------------------------------------------------------------------------------
Clock  Clock  Gating                # of   # of    # of       Toggle    Clock       Receiver
       Gate   Element               Fan    Clock   Registers  Savings   Gating      Clock
       Stage                        Out    Gates              (%)       Efficiency  Gates
-------------------------------------------------------------------------------------------------------------
CLK    1      clk_gate_R1_reg/latch  4     0       4          81.8%      27.3%
       1      clk_gate_Q_reg/latch   4     0       4          55.2%       3.4%
       2      clk_gate_R2_reg/latch  5     1       4          81.8%      54.5%     clk_gate_Q_reg/lat
-------------------------------------------------------------------------------------------------------------

1
```

The following example shows the hierarchical report for clock gates.

```
pt_shell> report_clock_gate_savings -by_clock_gate -hierarchical

****************************************
Report : Clock Gate Savings
Design : mydesign
         power_mode: Averaged
         -hierarchical
 ...
```

```
****************************************
-----------------------------------------------
Module              Number         Toggle
                 Clock Gates     Savings (%)
-----------------------------------------------
TOP                   225           67%
/A                    125           50%
//B                    30           12%
/C                    100           80%
-----------------------------------------------
1
```

The following report shows individual registers (not register banks). The clk/q ratio of 4 is expected for a register clocked every cycle, with random data on the d pin of the register.

```
pt_shell> report_clock_gate_savings -sequential

****************************************
Report : Clock Gate Savings
Design : mydesign
         power_mode: Averaged
         -sequential
 ...
****************************************
-------------------------------------------------------------------------------
Register      Root Clock      Local Clock    Q           Q/Clk       Register
              Toggle          Toggle         Toggle      Toggle      Gating
              Rate            Rate           Rate        Ratio       Efficiency
-------------------------------------------------------------------------------
TOP/A/U34     0.2             0.1          0.025         25%         50%
TOP/A/U35     0.2             0.1          0.01          10%         50%
-------------------------------------------------------------------------------
1
```

The following example shows the sequential hierarchical report, which averages over registers in each hierarchical block.

```
pt_shell> report_clock_gate_savings -sequential -hierarchical

****************************************
Report : Clock Gate Savings
Design : mydesign
         power_mode: Averaged
         -sequential
         -hierarchical
 ...
****************************************
---------------------------------------------------------------
Module     Number of          Q/Clk              Avg Register
           Registers          Toggle             Gating
                              Ratio              Efficiency
---------------------------------------------------------------
TOP        2250               17.5%              40.6%
 A         1250               15.6%              37%
 B          300               23.3%              70%
 C         1000               15.6%              45%
---------------------------------------------------------------
1
```

SEE ALSO

```
read_saif(2)
read_vcd(2)
set_switching_activity(2)
update_power(2)
```

report_clock_gating_check

Displays clock gating check information about specified pins.

SYNTAX

```
status report_clock_gating_check
   [-significant_digits digits]
   [-nosplit]
   [object_list]
```

Data Types

```
digits         integer
object_list    list
```

ARGUMENTS

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default.

-nosplit

Prevents line splitting and facilitates scripting to extract information from the report output.

object_list

Specifies a list of cells, pins, clocks, or design objects to report.

DESCRIPTION

The **report_clock_gating_check** command displays the following information for the clock gating checks. Information displayed about the specified objects: Instance of the cell, enable pin, clock pin, setup/hold time for rise, setup/hold time for fall, user-specified high/low active waveform. The asterisk (*) in the high/low field means that the user-specified high/low overrides what PrimeTime inferred from the logic of the cell. The following attributes show where the clock gating check originally was defined:

- I - Automatically inferred by PrimeTime.
- P - Power Compiler inserted the check.

- L - Defined by library cell.

EXAMPLES

The following example displays how to report all clock gating check informations.

```
pt_shell> report_clock_gating_check

****************************************
Report : clock gating check
Design : flop123
 ...
****************************************

                                  Rise                Fall
Cell     Enable  Clock     Setup     Hold      Setup    Hold   High/Low  Attr
-------------------------------------------------------------------------------
mid      A       B         5.00      3.00      5.00     3.00   High      I
MX       A       S         5.00      3.00      5.00     3.00   High      I
MX       B       S         5.00      3.00      5.00     3.00   Low (*)   I

Note: * indicates user override of tool inferred controlling value
Attr: I:auto inferred, P:power compiler inserted, L:library cell defined
```

SEE ALSO

```
report_constraint(2)
report_timing(2)
set_clock_gating_check(2)
report_default_significant_digits(3)
```

report_clock_timing

Reports timing attributes of clock networks.

SYNTAX

```
string report_clock_timing
  -type report_type
  [-clock clock_list]
  [-from_clock from_clock_list]
  [-to_clock to_clock_list]
  [-from from_list]
  [-to to_list]
  [-setup | -hold]
  [-launch | -capture]
  [-rise | -fall]
  [-nworst worst_entries]
  [-greater_than lower_limit]
  [-lesser_than upper_limit]
  [-slack_lesser_than slack_upper_limit]
  [-include_uncertainty_in_skew]
  [-verbose]
  [-significant_digits digits]
  [-show_clocks]
  [-crosstalk_delta]
  [-derate]
  [-variation]
  [-sort_by sort_attr]
  [-clock_crossing]
  [-include_clock_gating]
  [-nosplit]
  [-pre_commands pre_command_string]
  [-post_commands post_command_string]
```

Data Types

report_type	string
clock_list	list
from_clock_list	list
to_clock_list	list
from_list	list
to_list	list
worst_entries	int
lower_limit	float
upper_limit	float
slack_upper_limit	float
digits	int
sort_attr	string
pre_command_string	string
post_command_string	string

ARGUMENTS

-type *report_type*

Specifies the type of report to be generated. You can specify the following values for *report_type*:

- **transition** - Transition time report.
- **latency** - Latency report.
- **skew** - Skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, and **-lesser_than** options if you specify a skew report, and you can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew, or summary report.
- **interclock_skew** - Interclock skew report. You cannot use the **-launch**, **-capture**, **-rise**, **-fall**, and **-lesser_than** options if you specify an interclock skew report, and you can use the **-include_uncertainty_in_skew** option only in a skew, interclock_skew or summary report.
- **summary** - Summary report, which shows the worst instances of transition time, latency and skew over the clock networks or subnetworks of interest. If you specify a summary report, you can use only the **-clock**, **-to_list**, **-from_list**, **-include_uncertainty_in_skew**, **-nosplit**, and **-significant_digits** options.

  For nonsummary reports, report entries are ordered with respect to the specified attribute of interest (transition time, latency, or skew). Note that all skews reported are "local" skews. For an explanation of local skew, see the DESCRIPTION section.

-clock *clock_list*

Uses the specified clock networks in the report. A subreport is typically produced for every clock in the *clock_list*, unless you additionally specify the *to_list*, *from_list*, or both options that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *clock_list* and also issues a warning. By default, if you do not specify *clock_list*, all clocks in the design that have associated clock networks are used in the report.

-from_clock *from_clock_list*

Uses the specified clock networks as from-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *from_clock_list*, unless you additionally specify the *from_list* option that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *from_clock_list* and also issues a warning. By default, if you do not specify the *from_clock_list* option, all clocks in the design that have associated clock networks are used as from-clocks in the report.

-to_clock *to_clock_list*

Uses the specified clock networks as to-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *to_clock_list*, unless you additionally specify a *to_list* that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *to_clock_list* and also issues a warning. By default, if you do not specify the *to_clock_list* option, all clocks in the design that have associated clock networks are used as to-clocks in the report.

-from *from_list*

Considers the specified sequential clock network pins or ports as from-pins in the current report. If any named pin is not the clock pin of a sequential device, PrimeTime replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list and a warning message is issued.

-to *to_list*

Considers the specified sequential clock network pins or ports as to-pins in the current report. If any named pin is not the clock pin of a sequential device, such as a sink pin, PrimeTime replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively; if the *to_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list; if neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively; if the *from_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list; if neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

-setup

Uses only the setup data path is to be used in the report. Also, the skews, latencies or transition times reported must correspond to those used by the **report_timing** command in the verification of setup constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, the **-setup** option is assumed. You cannot use this option in summary reports.

-hold

Uses only the hold data path in the report. Also, the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of hold constraints. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, the **-setup** option is assumed. You cannot use this option in summary reports.

-launch

Uses only pins that launch data in the report. Also, latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are launching data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* arguments. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, the **-launch** option is assumed. You cannot use this option in summary or skew reports.

-capture

Uses only pins that capture data in the report. Also, the latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are capturing data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* options. In all other reports, the **-launch** and **-capture** options are mutually exclusive. If neither option is specified, the **-launch** option is assumed. You cannot use this option in summary or skew reports.

-rise

Specifies that the active transition is a rising edge for sequential device clock pins in the current report. The **-rise** and **-fall** options are mutually exclusive; if neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

-fall

Specifies that the active transition is a falling edge for sequential device clock pins in the current report. **-rise** and **-fall** are mutually exclusive; if neither is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to

answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

-nworst *worst_entries*

Specifies the number of worst report entries to be reported per clock domain; the default is 1. Entries are sorted with respect to the attribute of interest (transition time, latency or skew) specified with the **-type** option of the **report_type** command.

The worst entries are those most likely to cause a violation. For example, if you request a latency report using **-setup** and **-capture**, the smallest *worst_entries* are listed, sorted in ascending order, because small capture latencies for setup paths are more likely to lead to a violation than large capture latencies. By contrast, for skew reports, the worst entries always correspond to the largest skews over the specified domain. You cannot use this option in summary reports.

-greater_than *lower_limit*

Shows only those entries with an attribute value (latency, transition time or skew) greater (more positive) than *lower_limit*. You cannot use this option in summary reports.

-lesser_than *upper_limit*

Shows only those entries with an attribute value (latency, transition time or skew) less (more negative) than *upper_limit*. You cannot use this option in summary or skew reports.

-slack_lesser_than *slack_upper_limit*

Shows only those entries with a slack value less (more negative) than *slack_upper_limit*. For skew report entries slack is the worst slack over all paths launched by the from-pin and captured by the to-pin. For transition time or latency report entries, slack is defined as the worst slack of all paths either launched by a transition at the sink pin (if the **-launch** option is used) or captured by a transition at the sink pin (if the **-capture** option is used).

Note that the use of this filter might greatly increase the runtime of this report. However, when used with the **-capture** option the effect on runtime should be small, since PrimeTime can make use of cached slack values to avoid expensive recomputations. You cannot use this option in summary reports.

-include_uncertainty_in_skew

Specifies that the user-defined uncertainty between the sequential devices in a launch/capture pair is to be considered a component of skew. You cannot use this option in summary or skew reports.

-verbose

Generates a more detailed report; instead of a single line per pin, verbose reports trace the entire source-to-sink path through a clock network to show how the final reported attribute (skew, latency or transition time) was accumulated over the course of the path. You cannot use this option in summary reports.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-show_clocks

Shows the launching and capturing clocks for every interclock skew entry in the report. This is useful if *from_clock_list* or *to_clock_list* contain more than one clock each. You cannot use this option in interclock skew reports.

-crosstalk_delta

Shows the delta delay and delta transition time in verbose reports. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the **set_annotated_delay -delta_only** and **set_annotated_transition -delta_only** commands. Note that the **-crosstalk_delta** option reports only the calculated or annotated deltas, it does not initiate crosstalk analysis. Only deltas on input pins are shown.

-derate

Shows the derating factors in the report. By default, derating factors are not displayed. Derating factors are only shown when you specify the **-verbose** option.

-variation

Shows variation information. Using this option is allowed only when detailed variation-aware clock analysis is enabled. Without **-variation**, only the quantile or mean value is displayed, as specified using the **variation_derived_scalar_attribute_mode** variable. With **-variation**, the quantile, mean and either the standard deviation (for skew reports) or sensitivity (for latency or transition reports) is displayed for both verbose and non-verbose reports. In verbose reports, additional statistical info regarding individual path increments appear in new columns, namely the sensitivity, mean and increment, where the increment refers either to the quantile or the effective delay depending on the **variation_report_timing_increment_format** variable.

-sort_by *sort_attr*

Sorts the entries in the report with respect to the specified variational attribute. This option can be used only with the **-summary** option and only when detailed variation-aware clock analysis is enabled. Note that this option only controls the ranking of entries and not their display. You can specify the following values for *sort_attr*:

- **quantile** - Ranks entries using their quantile values.
- **mean** - Ranks entries using their mean values.
- **sensitivity** - Ranks entries using their sensitivity values. This option cannot be used in a **skew** or **interclock_skew** report.

-clock_crossing

Indicates that only skews between interacting clock domains (i.e. clocks connected by at least one non-false data path) is included in the report. This option can only be used in interclock_skew reports.

-include_clock_gating

Include clock pins of clock-gating checks in the report. By default, only sequential devices that exercise a setup/hold check are reported by report_clock_timing. This option additionally allows clock-gating checks to be reported.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. If this option is not used, most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-pre_commands *pre_command_string*

This option is available only if you invoke PrimeTime with the **-multi_scenario** option. This option allows you to specify a string of commands to be executed in the slave context before the execution of the merged reporting command. Commands must be grouped using the ";" character. The maximum size of a command is 1000 chars.

-post_commands ***post_command_string***

This option is available only if you invoke PrimeTime with the **-multi_scenario** option. This option allows you to specify a string of commands to be executed in the slave context after the execution of the merged reporting commands. Commands are grouped using the ";" character. The maximum size of a command is 1000 chars.

DESCRIPTION

This command generates a report of clock timing information for the current design.

There are several reporting types provided, which allow you to examine skew, latency and transition time attributes of a given clock network or subnetwork at various levels of generality. By default, the report displays the values of these attributes only at sink pins (that is, the clock pins of sequential devices) of the clock network. You use the **-verbose** option to display source-to-sink path traces. If you specify several clock domains, the **report_clock_timing** command generates a separate subreport for each clock domain.

At the highest level of abstraction is the summary style report, which provides only a list of maxima and minima of the skew, latency, and transition time attributes over the given networks. At a lower level of abstraction are the transition, latency, and skew type reports, called list style reports, in which you can sort, filter and display the worst set of sink pins in the given network with respect to a single attribute of interest. For skew reports, each report entry is a pair of sink pins and their relative skew. For transition time or latency reports, each entry corresponds to a single sink pin. The lowest level of abstraction is provided by verbose mode, which replaces every sink pin in a list style report by a corresponding source-to-sink path trace.

In both summary and list style reports, the rightmost column is an attributed column. Corresponding to each sink pin, the set of character symbols in this column indicates the following:

```
Symbol    Meaning
------    -------
 r        Rising transition
 f        Falling transition
 p        Propagated clock to this pin
 i        Clock inversion to this pin
 -        Launching transition
 +        Capturing transition
 m        Library clock insertion delay
```

In verbose mode, back-annotations on path elements in the timing path are indicated using a character symbol. For definitions of these character symbols, see the manual page of the **report_timing** command.

Skews reported by **report_clock_timing** are local skews only. Local skew exists from one sink pin to another only if their associated sequential devices are connected via a data path in the appropriate from-to sense. Note that even if all data paths between the sequential devices are false because of user-defined exceptions, the skew still qualifies as local skew if the devices are connected topologically.

If detailed variation-aware clock analysis is enabled using the **variation_analysis_mode** variable, the **report_clock_timing** command evaluates the attributes of latency, transition time and skew as statistical entities. By default, the quantile of these attributes is used for both ranking and display purposes. The ranking can be controlled using the **-sort_by** option, while the display parameter can be changed from quantile to mean using the **variation_derived_scalar_attribute_mode** variable. To augment the display with additional statistical parameters of **report_clock_timing** attributes, see the **-variation** option.

EXAMPLES

The following example shows a typical summary style report.

```
pt_shell> report_clock_timing -type summary

****************************************
Report : clock timing
        -type summary
        -nworst 1
 ...
****************************************

  Clock: CK1
-------------------------------------------------------------------------------
  Maximum setup launch latency:
      flop9/CP                                                 3.08      rp-+

  Minimum setup capture latency:
      or2_3/B                                                  1.15      fpi-+

  Minimum hold launch latency:
      or2_3/B                                                  1.15      fpi-+

  Maximum hold capture latency:
      flop9/CP                                                 3.08      rp-+

  Maximum active transition:
      or2_3/B                                                  0.20      fpi-+

  Minimum active transition:
      or2_3/B                                                  0.09      fpi-+

  Maximum setup skew:
      flop9/CP                                                           rp-+
      flop10/CP                                                0.87      rp-+

  Maximum hold skew:
      flop9/CP                                                           rp-+
      flop10/CP                                               -0.21      rp-+
-------------------------------------------------------------------------------
```

The following example displays the five worst setup skews in the clock network of CLK1, taking uncertainty into account.

```
pt_shell> report_clock_timing -clock CLK1 -type skew -setup \
             -nworst 5 -include_uncertainty_in_skew

****************************************
Report : clock timing
        -type skew
        -nworst 5
        -setup
        -include_uncertainty_in_skew
Design : multi_domain
 ...
****************************************

  Clock: CLK1
```

```
  Clock Pin                                Latency   Uncert    Skew
--------------------------------------------------------------------------------
  f2_2/CP                                  6.11                          rp-+
  f2_1/CP                                  2.01      0.11      4.21      fp-+

  l3_2/G                                   4.10                          rpi-
  f1_2/CP                                  1.00      0.22      3.32      rpi-+

  l2_2/G                                   4.11                          rp-
  f1_2/CP                                  1.00      0.12      3.23      rpi-+

  f2_2/CP                                  6.11                          rp-+
  l3_3/G                                   3.01      0.11      3.21      rp-

  l1_3/G                                   5.11                          rp-
  f2_1/CP                                  2.01      0.11      3.21      fp-+
--------------------------------------------------------------------------------
```

The following displays the five worst launching latencies for hold paths in the clock network of CLK2.

```
pt_shell> report_clock_timing -clock CLK2 -hold -launch -nworst 5 \
            -type latency

****************************************
Report : clock timing
        -type latency
        -launch
        -nworst 5
        -hold
Design : multi_domain
 ...
****************************************

  Clock: CLK2

                                                 --- Latency ---
  Clock Pin                      Trans     Source    Network   Total
--------------------------------------------------------------------------------
  f1_2/CP                        0.04      0.00      1.00      1.00      rpi-+
  f1_3/CP                        0.00      0.01      1.00      1.01      fp-+
  f2_1/CP                        0.01      0.01      2.00      2.01      rp-+
  f1_1/CP                        0.00      0.00      3.00      3.00      rpi-+
  f3_2/CP                        0.00      0.00      3.00      3.00      fpi-+
--------------------------------------------------------------------------------
```

The following example demonstrates how to request a verbose report showing the worst local skew from f2_2/CP to any other sink pin.

```
pt_shell> report_clock_timing -type skew -verbose -from f2_2/CP

****************************************
Report : clock timing
        -type skew
        -verbose
        -nworst 1
        -setup
Design : multi_domain
 ...
****************************************

  Clock: CLK1

  Startpoint: f2_2 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: f2_1 (rising edge-triggered flip-flop clocked by CLK1)
```

```
  Point                                Trans       Incr       Path
  -----------------------------------------------------------------
  clock source latency                             0.11       0.11
  clk2 (in)                             0.00       0.00       0.11 r
  az_1/Z (B1I)                          0.09       1.00 H     1.11 r
  az_2/Z (B1I)                          0.13       1.00 H     2.11 r
  bf2_2_1/Z (B1I)                       0.02       1.00 H     3.11 r
  if2_2_1/Z (IVA)                       0.44       1.00 H     4.11 f
  bf2_2_2/Z (B1I)                       0.01       1.00 H     5.11 f
  if2_2_2/Z (IVA)                       0.13       1.00 H     6.11 r
  f2_2/CP (FD1)                         0.13       0.00       6.11 r
  startpoint clock latency                                    6.11

  clock source latency                             0.01       0.01
  clk2 (in)                             0.00       0.00       0.01 r
  az_1/Z (B1I)                          0.02       1.00 H     1.01 r
  az_3/Z (B1I)                          0.01       1.00 H     2.01 r
  f2_1/CP (FD1)                         0.01       0.00       2.01 r
  endpoint clock latency                                      2.01
  -----------------------------------------------------------------
  startpoint clock latency                                    6.11
  endpoint clock latency                                     -2.01
  -----------------------------------------------------------------
  skew                                                        4.10
```

The following example traces the two worst launch latencies for hold paths in the clock network of CLK1.

```
pt_shell> report_clock_timing -type latency -hold -verbose \
            -nworst 2 -clock CLK1

****************************************
Report : clock timing
        -type latency
        -verbose
        -launch
        -nworst 2
        -hold
Design : multi_domain
 ...
****************************************

  Clock: CLK1

  Endpoint: f1_2 (rising edge-triggered flip-flop clocked by CLK1')

  Point                                Trans       Incr       Path
  -----------------------------------------------------------------
  clock source latency                             0.00       0.00
  clk1 (in)                             0.00       0.00       0.00 f
  if1_2_1/Z (IVA)                       0.04       1.00 H     1.00 r
  f1_2/CP (FD1)                         0.04       0.00       1.00 r
  total clock latency                                         1.00

  Endpoint: f1_3 (rising edge-triggered flip-flop clocked by CLK1)

  Point                                Trans       Incr       Path
  -----------------------------------------------------------------
  clock source latency                             0.01       0.01
  clk1 (in)                             0.00       0.00       0.01 r
  bf1_3_1/Z (B1I)                       0.00       1.00 H     1.01 r
  f1_3/CP (FD1)                         0.00       0.00       1.01 r
  total clock latency                                         1.01
```

The following example makes use of a slack filter to display the worst three skews between latches whose latch-to-latch paths are violating.

```
pt_shell> report_clock_timing -type skew -nworst 3 \
             -slack_lesser_than 0.0

****************************************
Report : clock timing
        -type skew
        -slack_lesser_than 0.00
        -nworst 3
        -setup
Design : multi_domain
 ...
****************************************

  Clock: CLKA

  Clock Pin                    Latency    CRP      Skew      Slack
--------------------------------------------------------------------------
  f2_2/CP                      6.01                                    rp-+
  f2_1/CP                      2.11      -0.03     3.87      -1.49     rp-+

  l2_2/G                       4.01                                    rp-
  f1_2/CP                      1.10      -0.21     2.70      -6.64     rpi-+

  l1_3/G                       5.01                                    rp-
  f2_1/CP                      2.11      -0.32     2.58      -1.63     rp-+
--------------------------------------------------------------------------
```

The following example requests the two largest setup skews for paths both launched and captured by any clock networks in the list *$my_clocks*.

```
pt_shell> report_clock_timing -type interclock_skew \
             -from_clock $my_clocks -to_clock $my_clocks \
             -nworst 2 -include_uncertainty_in_skew -show_clocks

****************************************
Report : clock timing
        -type interclock_skew
        -nworst 2
        -setup
        -include_uncertainty_in_skew
        -show_clocks
 ...
****************************************

  Number of startpoint pins   : 907
  Number of endpoint pins     : 907
  Number of startpoint clocks : 5
  Number of endpoint clocks   : 5

  Clock Pin                               Latency   Uncert    Skew
--------------------------------------------------------------------------
  orig_clk
  orig_if/ram_pdp1/FFB35/CK               1016.72                       rp-+
  dcd_ram/ram_clk
  dcd_ram/dcd_ram/RAM/CKRB                149.60    195.00    1062.12   rp-+

  comp_clk
  comp_if/c_port_if/edge_trig_reg/CK      1400.42                       rp-+
  comp_clk
  comp_if/c_port_if/posi/iq_reg/CK        1159.09   199.00    440.34    rp-+
--------------------------------------------------------------------------
```

The following reports the largest two setup skews in the domain of CLKB in the context of a detailed variation-aware clock analysis, and requests the augmentation of the report with variation information.

```
pt_shell> report_clock_timing -type skew -clock CLKB -variation

****************************************
Report : clock timing
        -type skew
        -nworst 2
        -variation
Design : multi_domain
 ...
****************************************

  Clock: CLKB

                                                --- Skew ---
  Clock Pin                    Latency   Stddev    Mean      Quant
-----------------------------------------------------------------------------
  f2_2/CP                      0.40221                                   rp-+
  f2_1/CP                      0.13533   0.02825   0.26688   0.34732   rp-+

  l2_1/CP                      0.33036                                   rpi-+
  l2_2/CP                      0.18638   0.02626   0.14398   0.20948   rp-+
-----------------------------------------------------------------------------
```

SEE ALSO

```
report_clock(2)
report_timing(2)
set_clock_uncertainty(2)
report_default_significant_digits(3)
timing_remove_clock_reconvergence_pessimism(3)
```

report_constraint

Displays constraint-related information about a design.

SYNTAX

```
status report_constraint
   [-all_violators]
   [-verbose]
   [-path_type slack_only | end]
   [-ignore_register_feedback feedback_slack_cutoff]
   [-max_delay]
   [-min_delay]
   [-max_capacitance]
   [-min_capacitance]
   [-max_transition]
   [-pulse_clock_max_transition]
   [-min_transition]
   [-pulse_clock_min_transition]
   [-max_fanout]
   [-min_fanout]
   [-min_pulse_width]
   [-pulse_clock_min_width]
   [-pulse_clock_max_width]
   [-min_period]
   [-recovery]
   [-removal]
   [-max_skew]
   [-clock_separation]
   [-clock_gating_setup]
   [-clock_gating_hold]
   [-significant_digits digits]
   [-nosplit]
   [-connection_class]
   [-boundary_check]
   [-boundary_check_include boundary_checks]
   [-pre_commands pre_command_string]
   [-post_commands post_command_string]
   [-format style]
   [-output file_name]
   [-default_scenario_name scen_name]
   [-waveform_integrity static | dynamic]
   [-pba_mode none | path | exhaustive]
   [-include_hierarchical_pins]
   [object_list]
```

Data Types

feedback_slack_cutoff	float
digits	integer
boundary_checks	list
pre_command_string	string
post_command_string	string
style	string

```
file_name                 string
scen_name                 string
object_list               list
```

ARGUMENTS

-all_violators

Shows a summary of the worst violation per endpoint of each violated design rule constraint in the current design.

-verbose

Shows detailed information about each constraint violation. Multiple violations for a given constraint are listed from the largest to the smallest violation.

-path_type slack_only | end

Specifies one of the following formats for the path report:

- **slack_only** (the default) - Shows only endpoint slacks.
- **end** - Shows the endpoint path total, required time, and slack for each path.

You cannot use the **-path_type** option together with the **-verbose** option.

-ignore_register_feedback *feedback_slack_cutoff*

Ignores any timing path that starts and ends at the same register and holds a value. This option applies to minimum delay as well as maximum delay reports. Only paths with slack less than the specified *feedback_slack_cutoff* are ignored. This option is applied as a filter to the paths after they are generated. Therefore, the number of paths generated might be less than the number specified with the **-nworst** and **-max_paths** options.

-max_delay

Shows only the maximum delay and setup information. In the default report, the worst violation per clock group is displayed.

The maximum delay information shows cost by path group. This includes violations of setup time on registers or ports with output delay as well as violations of **set_max_delay** commands. The total maximum delay cost is the sum of the weighted cost of each group. For information about creating path groups, see the **group_path** man page. To see the current path groups in the design, use the **report_path_group** command.

-min_delay

Shows only the minimum delay and hold information. In the default report, the sum of all violations per clock group is displayed.

The minimum delay cost includes violation of hold time on registers or ports with output delay as well as violations of the **set_min_delay** command.

-max_capacitance

Shows only the maximum capacitance constraint information. The **max_capacitance** design rule constraint limits the total capacitance on a net. The **-max_capacitance** option displays the **max_capacitance** cost (the sum of all maximum capacitance violations). To see details about the worst violation, use the **-verbose** and **-**

max_capacitance options together. To see details about all **max_capacitance** violations, use the **-all_violators**, **-verbose**, and **-max_capacitance** options together. Note that the **max_capacitance** constraint is not checked for load pins.

-min_capacitance

Shows only the minimum capacitance constraint information. The **min_capacitance** design rule constraint limits the total capacitance on a net.

-max_transition

Shows only the maximum transition constraint information. The **max_transition** design rule constraint limits the transition time on pins and ports. If the library uses the cmos2 delay model, maximum edge rate information is shown instead.

-pulse_clock_max_transition

Shows only the pulse clock maximum transition constraint information, similar to the **report_pulse_clock_max_transition** command.

-min_transition

Shows only the minimum transition constraint information. The **min_transition** design rule constraint specifies the minimum transition time on pins and ports. If the library uses the cmos2 delay model, maximum edge rate information is shown instead.

-pulse_clock_min_transition

Shows only the pulse clock minimum transition constraint information, similar to the **report_pulse_clock_min_transition** command.

-max_fanout

Shows only the maximum fanout constraint information. The **max_fanout** design rule constraint limits the fanout load on a net.

-min_fanout

Shows only the minimum fanout constraint information. The **min_fanout** design rule constraint specifies the minimum fanout load on a net.

-min_pulse_width

Shows only the minimum pulse width constraint information, similar to the **report_min_pulse_width** command. The **min_pulse_width** design rule constraint sets a minimum high or low pulse width at a clock pin or at pins or ports in the clock network.

-pulse_clock_min_width

Shows only the pulse clock minimum width constraint information. To enable pulse clock constraint checking, set the **timing_enable_pulse_clock_constraints** variable to **true**.

-pulse_clock_max_width

Shows only the pulse clock maximum width constraint information, similar to the **report_pulse_clock_max_width** command. To enable pulse clock constraint checking, set the **timing_enable_pulse_clock_constraints** variable to **true**.

-min_period

Shows only the minimum period constraint information. The **-min_period** option is a design rule used to set a minimum period on a clock signal.

-recovery

Shows only the recovery constraint information. The **-recovery** option is a timing constraint used to describe the minimum allowable time between the control pin transition to the inactive state, and the active edge of the synchronous clock signal. This time is from the control signal going inactive to the clock edge that latches data in. The asynchronous control signal must remain constant during this time, or an incorrect value might appear at the outputs. In the default report, the worst violation in the design is displayed.

-removal

Shows only the removal constraint information. The **-removal** option is a timing constraint used to describe the minimum allowable time between the clock pin inactive edge, while the asynchronous pin is active, to the inactive edge of the same asynchronous control pin. In the default report, the sum of all violations in the design is displayed.

-max_skew

Shows only the maximum skew constraint information. The **-max_skew** option is a timing constraint that checks the maximum separation time allowed between two clock signals.

-clock_separation

Shows only the clock separation constraint information.. The **-clock_separation** option is a timing constraint that checks the minimum separation time allowed between two clock signals.

-clock_gating_setup

Shows only the clock gating setup constraint information. The **-clock_gating_setup** option is a timing constraint used to set a minimum setup time between a clock and a signal controlling the gating of that clock. In the default report, the worst violation in the design is displayed.

-clock_gating_hold

Shows only the clock gating hold constraint information. The **-clock_gating_hold** option is a timing constraint used to set a minimum hold time between a clock and a signal controlling the gating of that clock. In the default report, the sum of all violations in the design is displayed.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option if you want to override the default.

-nosplit

Prevents line splitting to facilitate parsing of information from the report output.

-connection_class

Shows only the connection class constraint information. The connection_class constraint is displayed only if there is a connection_class violation. With the **-verbose** option, net pins and their associated connection class information are also displayed.

-boundary_check

Shows only HyperScale boundary constraint information. For more information, see the "Options for the HyperScale Flow" section.

-boundary_check_include *boundary_checks*

Reports the specified list of boundary checks. For more information, see the "Options for the HyperScale Flow"

section.

-pre_commands ***pre_command_string***

Specifies a string of commands to be executed in the slave context before the execution of merged reporting commands. Delimit commands with a semicolon (;). The maximum length of a command is 1000 characters. This option is available only if you invoke PrimeTime with the **-multi_scenario** option.

-post_commands ***post_command_string***

Specifies a string of commands to be executed in the slave context after the execution of merged reporting commands. Delimit commands with a semicolon (;). The maximum length of a command is 1000 characters. This option is available only if you invoke PrimeTime with the **-multi_scenario** option.

-format ***style***

Specifies the reporting format. The only allowed value is **csv**.

-output ***file_name***

Specifies the output file name for CSV report.

-default_scenario_name ***scen_name***

Specifies a scenario name in the CSV report. This option is valid only with **-format csv** and **-output** options. Note that in DMSA slaves, this option overrides the implicit slave scenario name by the provided scenario name, in the CSV reports.

-waveform_integrity static | dynamic

Specifies the type of waveform integrity analysis:

- **static** - Available only if advanced waveform propagation is enabled.
- **dynamic** - Available only if signal integrity analysis is enabled.

-pba_mode none | path | exhaustive

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.

To specify the constraints for waveform integrity analysis, use the **set_waveform_integrity_analysis** command.

-include_hierarchical_pins

Includes hierarchical pins in the timing report. This option is valid only when -verbose option is set to TRUE.

object_list

Specifies a list of pins, ports, or cells in the current design for which constraint related information is displayed. Note that pins and ports can only be specified with the **-max_transition** or **-min_capacitance** option. Similarly, cells can only be specified with the **-boundary_check** option. The **-verbose** option can be used to report detailed constraint calculation, and for the **-boundary_check** option, the **-all_violators** option can be used to report on all violating boundary checks in a HyperScale flow for a subset of instances.

DESCRIPTION

By default, the **report_constraint** command displays the following information about all timing and design rule constraints checks on the current design:

- Whether the constraint is violated or met
- By how much the constraint value is violated or met
- The design object that is the worst violation

The *object_list* can be optionally specified only with the **-max_transition** and **-min_capacitance** options. You have the flexibility of using the **-verbose** option to report detailed constraint calculation for the objects specified in the object list.

Options for the HyperScale Flow

If you enable HyperScale analysis by setting the **hier_enable_analysis** variable to **true**, you can report boundary constraint information during top-level HyperScale by using the following options:

-boundary_check

Shows HyperScale boundary constraint differences between the top and block levels that cannot be automatically resolved by the tool (non_auto_fixable violations). If there is an asterisk (*) beside the constraint name in the summary output of the report, the tool automatically generates updated boundary constraints to resolve these violation.

If you use the **-boundary_check** and **-verbose** options together, the command shows only the first violation for the non_auto_fixable type. To show all violations, use the **-all_violators** option.

-boundary_check_include *boundary_checks*

Reports the specified list of *boundary_checks*, which can contain the following values:

```
all
auto_fixable
boundary_ideal_network
boundary_logic_value
clock_attributes
clock_latency
clock_mapping
clock_relations
clock_skew_with_uncertainty
clock_uncertainty
data_arrival
env_variables
global_timing_derate
input_slews
library_mapping
non_auto_fixable
operating_conditions
```

For more information about a boundary check and how to resolve its violations, see the man page with "_violations" appended to the check name. For example, the following command shows information about **auto_fixable** violations:

```
pt_shell> man auto_fixable_violations
```

EXAMPLES

The following example shows brief constraint information for the current design:

```
pt_shell> report_constraint

****************************************
Report : constraint
Design : counter
 ...
****************************************

                                                    Weighted
    Group (max_delay/setup)      Cost     Weight     Cost
    -------------------------------------------------------
    CLK                          0.00      1.00      0.00
    default                      0.00      1.00      0.00
    -------------------------------------------------------
    max_delay/setup                                  0.00


    Constraint                                       Cost
    -------------------------------------------------------
    max_delay/setup                                  0.00 (MET)
    min_delay/hold                                   0.40 (VIOLATED)
    max_transition                                   0.00 (MET)
    max_fanout                                       0.00 (MET)
```

The following example displays detailed constraint information for the current design:

```
pt_shell> report_constraint -verbose

****************************************
Report : constraint
        -verbose
Design : counter
 ...
****************************************

  Startpoint: ffb (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)
  Path Group: CLK
  Path Type: max

  Point                                     Incr       Path
  -----------------------------------------------------------
  clock CLK (rise edge)                     0.00       0.00
  startpoint clock skew (ideal)             0.00       0.00
  startpoint clock uncertainty              0.00       0.00
  ffb/CP (FD3)                              0.00       0.00 r
  ffb/QN (FD3)                              2.42       2.42 r
  w/Z (ND4)                                 0.59       3.01 f
  q/Z (EO)                                  1.13       4.14 f
  j/Z (AO2)                                 1.08       5.22 r
  ffd/D (FDS2)                              0.00       5.22 r
  data arrival time                                    5.22

  clock CLK (rise edge)                    10.00      10.00
  endpoint clock skew (ideal)               0.00      10.00
  endpoint clock uncertainty                0.00      10.00
```

```
  ffd/CP (FDS2)                            0.00      10.00 r
  library setup time                      -0.90       9.10
  data required time                                  9.10
  -------------------------------------------------------------
  data required time                                  9.10
  data arrival time                                  -5.22
  -------------------------------------------------------------
  slack (MET)                                         3.88


    Design: counter

    max_area              30.00
  - Current Area          78.00
  -------------------------------
    Slack                -48.00  (VIOLATED)
```

The following example displays detailed information on only the worst violation per endpoint for those constraints that have violations:

```
pt_shell> report_constraint -all_violators -verbose

****************************************
Report : constraint
         -all_violators
         -verbose
 ...
****************************************

  Startpoint: b (input port)
  Endpoint: z5 (output port)
  Path Group: default
  Path Type: max

  Point                                    Incr       Path
  -------------------------------------------------------------
  input external delay                     0.00       0.00 r
  b (in)                                   0.00       0.00 r
  U5/Z (IV)                                1.32       1.32 f
  U3/Z (NR2)                               3.35       4.67 r
  U18/Z (AO6)                              0.73       5.40 f
  U22/Z (AO4)                              1.42       6.82 r
  z5 (out)                                 0.00       6.82 r
  data arrival time                                   6.82

  max_delay                                6.50       6.50
  output external delay                    0.00       6.50
 data required time                                 6.50
  -------------------------------------------------------------
  data required time                                  6.50
  data arrival time                                  -6.82
  -------------------------------------------------------------
  slack (VIOLATED)                                   -0.32


  Startpoint: c (input port)
  Endpoint: z3 (output port)
  Path Group: default
  Path Type: max

  Point                                    Incr       Path
  -------------------------------------------------------------
  input external delay                     0.00       0.00 r
  c (in)                                   0.00       0.00 r
```

```
  U6/Z (IV)                                  1.34       1.34 f
  U2/Z (NR2)                                 3.35       4.69 r
  U15/Z (AO7)                                0.87       5.56 f
  U24/Z (AO3)                                1.02       6.57 r
  z3 (out)                                   0.00       6.57 r
  data arrival time                                     6.57

  max_delay                                  6.50       6.50
  output external delay                      0.00       6.50
  data required time                                    6.50
  -------------------------------------------------------------
  data required time                                    6.50
  data arrival time                                    -6.57
  -------------------------------------------------------------
  slack (VIOLATED)                                     -0.07


    Net: a

    max_fanout             5.00
  - Fanout                 7.00
  ------------------------------
    Slack                 -2.00  (VIOLATED)
```

The following example shows how to report all max transition violations:

```
pt_shell> report_constraint -max_transition -all_violators

****************************************
Report : constraint
        -all_violators
        -path_type slack_only
        -max_transition
 ...
****************************************

   max_transition

                        Required        Actual
   Pin                  Transition     Transition       Slack
   ---------------------------------------------------------------
   CK2                     0.10          10.00          -9.90  (VIOLATED)
   m0/B                    0.10          10.00          -9.90  (VIOLATED)
   fc/D                    0.10           1.75          -1.65  (VIOLATED)
```

The following example shows how to report all minimum pulse width violations:

```
pt_shell> report_constraint -min_pulse_width -all_violators

****************************************
Report : constraint
         -all_violators
         -min_pulse_width
 ...
****************************************

   sequential_clock_pulse_width

                         Required       Actual
   Pin                   pulse width    pulse width    Slack
   ------------------------------------------------------------------------
   ff1/CP                10.20          10.00          -0.20  (VIOLATED)
   ff2/CP                10.20          10.04          -0.16  (VIOLATED)
   ff3/CP                 2.10           2.00          -0.10  (VIOLATED)
```

```
   ff3/CP                 2.10           2.00          -0.10  (VIOLATED)
   ff2/CP                10.00           9.96          -0.04  (VIOLATED)


   clock_tree_pulse_width

                         Required       Actual
   Pin                   pulse width    pulse width    Slack
   ---------------------------------------------------------------------
   nand1/Z               10.00           9.58          -0.42  (VIOLATED)
   or1/B                 10.00           9.58          -0.42  (VIOLATED)
   nand1/A               10.20          10.00          -0.20  (VIOLATED)
   or1/Z                 10.20          10.04          -0.16  (VIOLATED)
   or1/Z                 10.00           9.96          -0.04  (VIOLATED)
```

The following example displays how to report all max_skew violations:

```
pt_shell> report_constraint -max_skew -all_violators

****************************************
Report : constraint
        -all_violators
        -path_type slack_only
        -max_skew
 ...
****************************************

   max_skew
                         Required       Actual
   Pin                   pulse width    pulse width    Slack
   ---------------------------------------------------------------------
   ff3/c (r->f)          0.15           7.46          -7.31  (VIOLATED)
   ff1/c (r->f)          0.15           5.47          -5.32  (VIOLATED)
   ff2/c (f->f)          0.14           2.32          -2.18  (VIOLATED)
   ff2/c (r->r)          0.12           1.18          -1.06  (VIOLATED)
   ff4/c (f->f)          0.14           0.84          -0.70  (VIOLATED)
   ff4/c (r->r)          0.12           0.42          -0.30  (VIOLATED)
```

The following example displays how to report detailed connection_class violations:

```
pt_shell> report_constraint -connection_class -verbose

****************************************
Report : constraint
        -verbose
 ...
****************************************

    Net: en

    Pin                                Connection_class
 ---------------------------------------------------------------------
                            User defined              Library defined
 ---------------------------------------------------------------------
   en                       xyz
   U18/B                                              default
   U20/B                                              default
   U22/B                                              default
   U14/B                                              default
   U16/A                                              default

    connection_class        0.00
  - Cost                    6.00
  -----------------------------
```

```
        Slack                  -6.00  (VIOLATED)
```

SEE ALSO

```
create_clock(2)
group_path(2)
report_clock(2)
report_design(2)
report_min_pulse_width(2)
report_path_group(2)
report_pulse_clock_max_transition(2)
report_pulse_clock_max_width(2)
report_pulse_clock_min_transition(2)
report_pulse_clock_min_width(2)
report_timing(2)
set_max_delay(2)
set_waveform_integrity_analysis(2)
auto_fixable_violations(3)
non_auto_fixable_violations(3)
report_default_significant_digits(3)
timing_enable_pulse_clock_constraints(3)
timing_max_capacitance_derate(3)
timing_max_transition_derate(3)
```

report_context

Reports the characterized timing context information.

SYNTAX

```
string report_context
   [-timing]
   [-environment]
   [-design_rules]
   [-constant_inputs]
   [-nosplit]
   [-multi_instance_merge_compatibility]
   [-list_context_override]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-timing

Reports timing information, such as clocks, input and output delays, and timing exceptions.

-environment

Reports environment related information, such as operating conditions (process, temperature and voltage), wire load model, capacitive loads on input and output pins, and driving cell information on input pins.

-design_rules

Reports characterized design rule information, such as *max_capacitance*, *max_transition*, and *max_fanout*.

-constant_inputs

Reports logic constants propagated to input pins.

-nosplit

Does not split lines if column overflows.

-multi_instance_merge_compatibility

Specifies clustering information within each multi-instance group that satisfies clock compatibility. This option can only be used for the HyperScale Constraint Extractor or characterize context flows. For multi-instance configurations, it checks if the clock constraints of different instances are compatible or not. If the clock constraints are not compatible, this means that those instances should not be put in one group. It provides

clustering information within each group that satisfies clock compatibility within each cluster. When this option is used, this information is printed as part of the report. A new column is added for this option which shows the cluster. A number shows the cluster number for each instance in that group. If the instance is not part of multi-instance configuration, -1 is show as the cluster.

-list_context_override

This option enables reporting of context overriding exceptions at HyperScale block level analysis. Automatic override of user specified IO budgets in the block constraints with accurate HyperScale top level context is critical for HyperScale flow to achive flat analysis accuracy at block level analysis. However, there are multiple reasons this automatic context overriding process maybe either be skipped, or only success partially, or completely fail.

Main reasons causing the overriding exceptions are:

1. user directive with **set_dont_override**;
2. error conditions such as clock mapping error
3. netlist change on port nets.

This report lists all the ports whose context override dod not happen completely successfully with the reason(s).

If there is no top level HyperScale context available for the current block level run, no ports will be automatically override and therefore no ports will be listed by this report. For such case, a message indicating there is no context loaded for current design will be issued.

The column 'Not Overridden Reason' will list the reasons for ports don't have context overridden while there is context available. The reasons include: dont_override_timing: port with user set don't override on timing dont_override_noise: port with user set don't override on noise net_mismatch: port with netlist mismatch on port net unresolved: port context refer to clock(s) with mapping errors

object_list

Lists cells or pins/ports for which to report the context.

DESCRIPTION

Reports the timing context captured using the **characterize_context** command for the specified list of instances or pins.

This command could also be used to report context loaded from read_context command. Use current_design as the object if reporting all loaded context. It also can be used to report context loaded on specified ports.

Options specify categories of context information to report. All information categories are reported if no option is specified.

EXAMPLES

The following command reports the timing-related context information for instance I1 and I2. The report shows clocks and their waveforms, point-to-point timing exception, input external delays, and output external delays.

The input and output delay information lists the minimum rise, minimum fall, maximum rise, and maximum fall

delays. Input and output delay information lists the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered as coming from a level-sensitive latch, or output delay is considered as going to a level-sensitive latch. The default is that the delay is relative to a rising edge-triggered device.

Multiple output delays are listed for a characterized output pin if it has more than one fanout or if the logic it drives terminates in more than one type of latches (that is, edge-triggered (falling or rising clock) or level-sensitive with falling or rising clock).

```
pt_shell> report_context -timing {I1 I2}
```

The following command reports the environment and design rule context information. The report shows design rule checking, wire load models, drive information of input pin, and capacitive load on input and output pins of I2.

```
pt_shell> report_context -env -design_rules I2
```

The following command reports the context on port "in" loaded from report_context.

```
pt_shell> report_context in
```

The following command reports the context loaded from report_context.

```
pt_shell> report_context [current_design]
```

SEE ALSO

characterize_context(2)
write_context(2)

report_crpr

Reports the clock reconvergence pessimism (CRP) calculated between specified register clock pins or ports.

SYNTAX

```
string report_crpr
   -from from_latch_clock_pin
   -to to_latch_clock_pin
   [-from_clock from_clock]
   [-to_clock to_clock]
   [-setup | -hold]
   [-significant_digits digits]
```

Data Types

```
from_latch_clock_pin     string
to_latch_clock_pin       string
from_clock               string
to_clock                 string
digits                   integer
```

ARGUMENTS

-from *from_latch_clock_pin*

Specifies the clock pin of the launching sequential device or clock gating check to be reported. A constrained input port can also be used.

-to *to_latch_clock_pin*

Specifies the clock pin of the capturing sequential device or clock gating check to be reported. A constrained output port can also be used.

-from_clock *from_clock*

Specifies the clock that fans out to the launching sequential device.

-to_clock *to_clock*

Specifies the clock that fans out to the capturing sequential device.

-setup

Indicates that the CRP used for a setup data path is to be reported. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-hold

Indicates that the CRP used for a hold data path is to be reported. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13.

If this option is not specified, the number of significant digits is determined by the **report_default_significant_digits** variable.

DESCRIPTION

Reports the clock reconvergence pessimism (CRP) calculated between specified register clock pins or ports. This command displays the following information about the calculation of the CRP between the two specified sequential elements:

- The name of the common pin in the clock network.
- The clock edges (rising or falling) that propagate through the common point to the launching and capturing registers.
- The value of the **timing_crpr_threshold_ps** variable.
- A tabulation of the arrival times for both clock edges at the common point and their associated CRP (crp_rise and crp_fall).
- The CRP used along with the reasoning behind the selection of the particular CRP (crp_rise or crp_fall) used for that report.
- The CRP value used is independent of the CRPR threshold, and, therefore, is referred to as the exact CRP. The **report_timing** and **report_clock_timing** commands report a CRP value that might be pessimistic by up to the value of the CRPR threshold. If the CRP value used by the **report_timing** or **report_clock_timing** command differs from the exact CRP, this value is also reported.

If the **timing_remove_clock_reconvergence_pessimism** variable is set to **false**, CRPR is inactive and this command does not work.

To generate a CRP report for specific clocks, use the **-from_clock** and **-to_clock** options. Otherwise, the tool reports all clocks that fanout to each sequential device.

For each potential common point in the design, the tool calculates

- crp_rise from rising arrival times at the common point
- crp_fall from falling arrival times at the common point

The value of CRP used in the report is dependent on what clock edges propagate through the common point to the clock pins of the launching and capturing devices. The clock edges are included in the CRPR report as "Launching edge at common point", "Capturing edge at common point".

For example, if a rising edge through the common point triggers the launching device and also triggers the capturing device, then a rising CRP value (crp_rise) is used. Similarly, for a falling edge propagating through the common point and subsequently launching and capturing data, a falling CRP value is used (crp_fall).

If a rising edge through the common point launches the data, and a falling edge through the common point captures it, a mismatch in sense occurs. In this case, if the **timing_clock_reconvergence_pessimism** variable is set to

- **normal**, the minimum of crp_rise and crp_fall is used
- **same_transition**, the CRP is reported as zero

This selection process is reported in the selection details section of the report.

If either clock edge is reported as "RISING or FALLING", there is a non-unate path between the common point and that clock pin, such that botch edge directions result in active clock edges at the clock pin.

If dynamic information is enabled, two sets of arrival totals are tabulated. Dynamic information can result from SI delta delays, dynamic clock latencies or dynamic rail voltages. In this case, the two sets of arrival totals tabulated are those computed with and without dynamic information being available. The arrival totals computed without dynamic information are equivalent to those used in a PrimeTime analysis with no dynamic information available.

The additional set of arrivals result in four CRP values from which the CRP value used should be chosen. These four CRP values can be summarized as rise/fall CRP and rise/fall dynamic CRP. The rise/fall decision is made based on clock edges incident at the clock edge as before. The decision over whether or not to use dynamic CRP is based on the temporal separation of launching and capturing clock edges. It is only valid to use dynamic CRP if the clock edge launching and capturing data occurs at the same time.

In the case when the capturing sequential device is a level-sensitive latch, two CRP values are reported. The first corresponding to the opening edge of the latch (this CRP also appears in the timing report) and the other corresponding to the closing edge of the device. The selection details section also reports the decision making process behind both CRP values. Note that in this case, since the opening and closing clock edges at the latch are always different, there is always a mismatch in clock edges for one of them.

EXAMPLES

The following example displays how to use the command to report the clock reconvergence pessimism calculated between the sequential elements, ffa and ffd, for a setup data path.

```
pt_shell> report_crpr -from ffa/CP -to ffd/CP -setup

report_crpr -from ffa/CP -to ffd/CP -setup
****************************************
Report : CRP Calculation
Design : counter
 ...
****************************************

  Startpoint: ffa (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)

  Common Point: CLK <inside>
  Common Clock: CLK
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times                      Early    Late     CRP
-------------------------------------------------------------
  Rise                                3.00    19.00    16.00
  Fall                                5.00    23.00    18.00
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Edge Match:          Match, using rise CRP
```

```
  ---------------------------------------------------------------
    clock reconvergence pessimism                        16.00
```

The following example displays a report where the launching device is a flip-flop and the capturing device is a latch. Note that in this case the mismatch in clock edges occurs for the CRP corresponding to the closing edge at the latch.

```
pt_shell> report_crpr -from reg1/CP -to lat2/G

****************************************
Report : CRP Calculation
Design : borrow
 ...
****************************************

  Startpoint: reg1 (rising edge-triggered flip-flop clocked by clk)
  Endpoint: lat2 (positive level-sensitive latch clocked by clk)

  Common Point: clk_buf/Z
  Common Clock: clk
  Launching edge at common point: RISING
  Latch open edge at common point: RISING
  CRPR threshold: 0.01

  Arrival Times                      Early    Late     CRP
  ---------------------------------------------------------------
  Rise                              2.1229   2.6529   0.5300
  Fall                              2.0868   2.7868   0.7000
  ---------------------------------------------------------------

  Selection Details
  ---------------------------------------------------------------
  Edge Match (opening): Match, using rise CRP
  Edge Match (closing): Mismatch, using min(rise CRP, fall CRP)
  ---------------------------------------------------------------
  clock reconvergence pessimism (open edge)           0.5300
  clock reconvergence pessimism (close edge)          0.5300
```

The following example displays the case when both SI and CRPR are enabled. As outlined in a previous paragraph, two sets of arrival totals are computed in this case (for example, when both CRPR and SI are enabled). One set of arrival totals are computed with no SI information and another are computed considering SI information. To illustrate this, both a setup and hold report are displayed to highlight the case when CRP is calculated from arrival times without delta delays (setup) and the case when CRP is calculated from arrival times including delta delays (hold with the launching and capturing clock edge occurring at the same time).

```
pt_shell> report_crpr -from seg1/u3/CK -to seg1/u9/CK

****************************************
Report : CRP Calculation
 ...
****************************************

  Startpoint: seg1/u3 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: seg1/u9 (rising edge-triggered flip-flop clocked by CLK1)

  Common Point: seg1/u17_c/Y
  Common Clock: CLK1
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times (Static)             Early    Late     CRP
  ---------------------------------------------------------------
```

```
  Rise                              0.3633   0.3746   0.0113
  Fall                              0.3871   0.3991   0.0120
-------------------------------------------------------------

  Arrival Times (Dynamic)            Early    Late     CRP
-------------------------------------------------------------
  Rise                              0.2142   0.3746   0.1604
  Fall                              0.2296   0.3991   0.1695
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Arrival Times:       Static
  Edge Match:          Match, using rise CRP
-------------------------------------------------------------
  clock reconvergence pessimism                       0.0113


  Range of accuracy of CRP in report_timing, due to value
  of timing_crpr_threshold_ps:    0.0000 <= CRP <= 0.0113


pt_shell> report_crpr -from seg1/u3/CK -to seg1/u9/CK -hold

****************************************
Report : CRP Calculation
 ...
****************************************

  Startpoint: seg1/u3 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: seg1/u9 (rising edge-triggered flip-flop clocked by CLK1)

  Common Point: seg1/u17_c/Y
  Common Clock: CLK1
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times (Static)             Early    Late     CRP
-------------------------------------------------------------
  Rise                              0.3633   0.3746   0.0113
  Fall                              0.3871   0.3991   0.0120
-------------------------------------------------------------

  Arrival Times (Dynamic)            Early    Late     CRP
-------------------------------------------------------------
  Rise                              0.2142   0.3746   0.1604
  Fall                              0.2296   0.3991   0.1695
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Arrival Times:       Static
  Edge Match:          Match, using rise CRP
-------------------------------------------------------------
  clock reconvergence pessimism                       0.1604
```

SEE ALSO

set_clock_latency(2)

```
set_rail_voltage(2)
si_enable_analysis(3)
timing_clock_reconvergence_pessimism(3)
timing_crpr_threshold_ps(3)
timing_remove_clock_reconvergence_pessimism(3)
```

report_current_calculation

Shows the current and intrinsic RC calculation process carried out by the **update_currents** command.

SYNTAX

```
status report_current_calculation
   object_list
   [-internal]
   [-verbose]
   [-fsdb_file file_name]
```

Data Types

```
list object_list
string file_name
```

ARGUMENTS

object_list

A list of cells on which to report the current and intrinsic RC calculation process.

-internal

Indicates to report static current for internal pg pins of hard macro for static rail analysis.

-verbose

Indicates to report more details.

-fsdb_file *file_name*

Specifies an output FSDB file to capture the current waveform for each event in dynamic analysis (**update_currents -dynamic**) if there is only one cell in the *object_list*.

DESCRIPTION

The command provides detailed current and intrinsic RC calculation information for the power and ground pins of all the cells specified by the cell list. The PG pin current and intrinsic RC used to perform rail analysis are derived from power analysis and library data when **update_currents** is executed. This report details how these values are derived from those inputs.

If the command **update_currents -static** is performed for static analysis, the report contains the static part

describe below. If the command **update_currents -dynamic** is performed for dynamic analysis, the report includes both the static and dynamic part.

For standard cells or macro cells without modeling data, the static part of the report includes the following information:

- A table reporting the units used throughout the report. The library units are used wherever applicable.
- A detailed report per specified cell, including:
  - Information for each signal pin of the cell, including:
    - Rise transition time (slew)
    - Fall transition time (slew)
    - Probability of a '1' state being on the pin
    - Toggle rate (toggles per time unit)
  - Information for each power pin of the cell, including:
    - Calculated power (P)
    - Ideal voltage (V)
    - Derived static current (P/V)
  - Information for each ground pin of the cell, including:
    - Contribution to the ground pin from each power pin, including:
      - Percent contribution of the ground pin power/current from the power pin
      - Power contributed by the power pin
      - Current contributed by the power pin
    - Total power for the ground pin (as summed from the power pin contributions)
    - Ideal voltage
    - Total static current (as summed from the power pin contributions)

For macro cells with modeling data (DWM-Lite), the static part of information reported is the same as for standard cells, with the exception of the detailed reports for the power and ground pins. The currents calculated for macro power or ground pins is based upon the operating mode information provided with the macro model. This report includes that information:

- Information for each power/ground pin of the cell, including:
  - Information on each defined operating mode (by mode name):
    - Triggering signal pin
    - Defined rise/fall direction for the trigger
    - When condition
    - Switching activity for the trigger pin and its direction (toggles per time unit)
    - Probability of the when condition being met

        - Charge consumed per trigger
        - Derived static current at the power pin
    - Calculate Power (P)
    - Ideal voltage (V)
    - Derived static current (summed from the operating mode information)

For standard cells or macro cells without modeling data, the dynamic part of the report includes the following information for each switching event:

- Event happening time
- Energy consumed
- Signal pin state
- Current waveform construction detail for power and ground pins

For macro cells with modeling data (DWM-Lite), the dynamic part of information reported is the same as for standard cells, with the exception of triggered macro modes. The triggered macro mode calculated for macro power or ground pins is based upon the operating mode information provided with the macro model. This report includes that information:

- When condition
- Trigger pin
- Macro mode activated
- Current waveform for the macro mode

EXAMPLES

The following example shows a sample report for standard cell INV6, which has three power pins and three ground pins, in static analysis:

```
pr_shell> report_current_calculation INV6

****************************************
Report : current calculation
Design : test_chip
Version: D-2010.06-Beta
Date   : Wed Apr 21 17:10:06 2010
****************************************

Units
----------------------------------------------
Capacitive_load_unit        : 1e-12 Farad
Current_unit                : 0.001 Amp
Resistance_unit             : 1000 Ohm
Time_unit                   : 1e-09 Second
Voltage_unit                : 1 Volt
Power_unit                  : 1 Watt
Charge_unit                 : 10-12 Coulomb

==================== Cell INV6, lib_cell typical_invt ====================
```

```
Input pin : i
  Rise transition     : 0.0869193
  Fall transition     : 0.0582622
  State probability   : 0.59
  Switching activity  : 0.148148

Output pin : zn
  Rise transition     : 0.314579
  Fall transition     : 0.18178
  Loading capacitance : 0.232085
  State probability   : 0.41
  Switching activity  : 0.148148

Power pin : vdd
  Power                  : 7.2e-05
  Voltage                : 1.2
  Static current         : -0.06 (-6e-05 Amp)

Power pin : vbp
  Power                  : 3.6e-05
  Voltage                : 1.2
  Static current         : -0.03 (-3e-05 Amp)

Power pin : vdp
  Power                  : 3.6e-05
  Voltage                : 1.2
  Static current         : -0.03 (-3e-05 Amp)



Ground pin : vss
  Its power and static current are calculated from power pins
    Power pin : vdd
      Allocation     : 33.3333%
      Power          : 2.4e-05
      Static current : 0.02 (2e-05 Amp)
    Power pin : vbp
      Allocation     : 33.3333%
      Power          : 1.2e-05
      Static current : 0.01 (1e-05 Amp)
    Power pin : vdp
      Allocation     : 33.3333%
      Power          : 1.2e-05
      Static current : 0.01 (1e-05 Amp)

  Power                  : 4.8e-05
  Voltage                : 0
  Static current         : 0.04 (4e-05 Amp)


Ground pin : vbn
  Its power and static current are calculated from power pins
    Power pin : vdd
      Allocation     : 33.3333%
      Power          : 2.4e-05
      Static current : 0.02 (2e-05 Amp)
    Power pin : vbp
      Allocation     : 33.3333%
      Power          : 1.2e-05
      Static current : 0.01 (1e-05 Amp)
    Power pin : vdp
      Allocation     : 33.3333%
      Power          : 1.2e-05
      Static current : 0.01 (1e-05 Amp)
```

```
    Power                  : 4.8e-05
    Voltage                : 0
    Static current         : 0.04 (4e-05 Amp)


  Ground pin : vdn
    Its power and static current are calculated from power pins
      Power pin : vdd
        Allocation     : 33.3333%
        Power          : 2.4e-05
        Static current : 0.02 (2e-05 Amp)
      Power pin : vbp
        Allocation     : 33.3333%
        Power          : 1.2e-05
        Static current : 0.01 (1e-05 Amp)
      Power pin : vdp
        Allocation     : 33.3333%
        Power          : 1.2e-05
        Static current : 0.01 (1e-05 Amp)

    Power                  : 4.8e-05
    Voltage                : 0
    Static current         : 0.04 (4e-05 Amp)
```

The following example shows a sample report for hard macro sram1056x12_instance_5_2 in static analysis:

```
pr_shell> report_current_calculation sram1056x12_instance_5_2

****************************************
Report : current calculation
Design : test_hm_chip
Version: D-2010.06-Beta
Date   : Fri Apr 23 15:10:25 2010
****************************************

Units
----------------------------------------------
Capacitive_load_unit        : 1e-12 Farad
Current_unit                : 1e-06 Amp
Resistance_unit             : 1000 Ohm
Time_unit                   : 1e-09 Second
Voltage_unit                : 1 Volt
Power_unit                  : 1 Watt

Charge_unit                 : 1e-15 Coulomb


==================== Cell sram1056x12_instance_5_2, lib_cell sram1056x12 ====================

  Input pin : CLK
    Rise transition     : 0
    Fall transition     : 0
    State probability   : 0.19
    Switching activity  : 0.0582453

  Input pin : CEN
    Rise transition     : 0.0432199
    Fall transition     : 0.0284086
    State probability   : 0.96
    Switching activity  : 0.00546238

  Input pin : WEN
    Rise transition     : 0.0278887
    Fall transition     : 0.0181914
```

```
    State probability   : 1
    Switching activity  : 0

  Power pin : VDD
    Mode name : Read
      Trigger signal pin          : CLK
      Trigger direction           : rise
      When condition              : ((!CEN)&WEN)
      Trigger activity            : 0.0291227
      When condition probability  : 0.04
      Charge per trigger          : 12128.5
      Static current              : -14.1285 (-1.41285e-05 Amp)
    Mode name : Write
      Trigger signal pin          : CLK
      Trigger direction           : rise
      When condition              : ((!CEN)&(!WEN))
      Trigger activity            : 0.0291227
      When condition probability  : 0
      Charge per trigger          : 15045.6
      Static current              : -0 (-0 Amp)

    Power                  : 2.11928e-05
    Voltage                : 1.5
    Static current         : -14.1285 (-1.41285e-05 Amp)


  Ground pin : VSS
    Mode name : Read
      Trigger signal pin          : CLK
      Trigger direction           : rise
      When condition              : ((!CEN)&WEN)
      Trigger activity            : 0.0291227
      When condition probability  : 0.04
      Charge per trigger          : 12128.5
      Static current              : 14.1285 (1.41285e-05 Amp)
    Mode name : Write
      Trigger signal pin          : CLK
      Trigger direction           : rise
      When condition              : ((!CEN)&(!WEN))
      Trigger activity            : 0.0291227
      When condition probability  : 0
      Charge per trigger          : 15045.6
      Static current              : 0 (0 Amp)

    Power                  : 2.11928e-05
    Voltage                : 0
    Static current         : 14.1285 (1.41285e-05 Amp)
```

SEE ALSO

update_currents(2)

report_delay_calculation

Displays the actual calculation of a cell or net timing arc delay value.

SYNTAX

```
int report_delay_calculation
   [-min | -max]
   [-from from_pin]
   [-to to_pin]
   [-of_objects objects
   [-nosplit]
   [-from_rise_transition value]
   [-from_fall_transition value]
   [-thresholds]
   [-crosstalk]
   [-derate]
```

Data Types

from_pin	string
to_pin	string
objects	collection
value	float

ARGUMENTS

-min

Shows the minimum delay calculation. The design must be in either min or max mode.

-max

Shows the maximum delay calculation. This is the default if neither the **-min** nor **-max** option is specified.

-from *from_pin*

Specifies the startpoint of a timing arc within a design. For a cell timing arc, the pins must represent the input and output pins of a common leaf cell that have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. Port names are allowed in place of pin names for net arcs. Use either the **-from** and **-to** options together, or use the **-of_objects** option. However, you cannot set both.

-to *to_pin*

Specifies the endpoint of a timing arc within a design. For a cell timing arc, the pins must represent the input and output pins of a common leaf cell that have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. Port names are allowed in place of pin names for net arcs. Use either the **-from** and **-to** options together, or use the **-of_objects** option. However, you cannot

set both.

-of_objects ***objects***

Specifies a collection of timing arcs (created with the **get_timing_arcs** command) on which to report. Arcs in the list are reported in order of from and to pins. Use either the **-from** and **-to** options together, or use the **-of_objects** option. However, you cannot set both.

-nosplit

Prevents line-splitting to facilitate parsing of information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-from_rise_transition ***value***

Specifies a value to be used by the delay calculation for the from rise transition.

-from_fall_transition ***value***

Specifies a value to be used by the delay calculation for the from fall transition.

-thresholds

Reports the characterization thresholds that are used for delay calculation.

-crosstalk

Reports the crosstalk information for a net arc. The arc is specified with the **-from_pin** and **-to_pin** options. It is not permitted with the **-of_objects** option and the user-chosen transition time **-from_rise_transition** and **-from_fall_transition** options. The crosstalk information is reported from the last **update_timing** command.

-derate

Reports derating components and derated delay.

DESCRIPTION

This command provides detailed timing calculation information about the specified cell or net timing arc. This information is useful for debugging or verifying timing data in a technology library.

Both operating conditions and wire load models are considered when making delay calculations. Timing ranges as affected by the **set_timing_derate** command (similar to the **set_timing_range** dc_shell command) are not considered because they typically apply to an entire path.

The information reported is solely for the input arcs. Even if there are other arcs merging at the to-pin of an input arc, the transition time reported for the to-pin is the transition time that the input arc contributes to the merging point. As a result, the reported transition time might not match the transition time at the to-pin because the final transition time is the result of merging transition times contributed by all the input arcs. To get the final transition time at the to-pin, either all these input arcs should be reported or an attribute access to the to-pin should be used.

The **-crosstalk** option on net arc, reports the aggressor and victim information for both rise and fall. It prints coupling capacitance, the driving library cell, the clocks reaching the net (both victim and aggressor). The aggressors have "Switching Bump", which is used for prioritizing the aggressors, and also have aggressor attributes. These informations are already available as net or design attributes. For example, "Switching Bump" is available as the net attribute si_xtalk_bumps. Considering the memory usage, only the worst case bumps for the

net is stored, even though during the crosstalk delay calculation switching bump per net arc is used. "Switching Bump" is crosstalk delay bump ratio of VDD. The clocks gives insight into the constraints of the coupling cluster. For example, during the normal mode analysis the test clocks are supposed to be quiet. If there is no arrival windows, it reported as "No Arrival". Driving cell gives a idea relative strength of the victim and aggressor drivers. If the ports/pins have no driving cell it is reported as "No Lib Cell".

It also prints the other settings in design that could effect the crosstalk delay calculation, for example, logical correlation, composite aggressor setting, crosstalk delay analysis mode, and crosstalk delay effort level.

Attributes:

```
A - aggressor is Active
C - aggressor is a composite aggressor
E - aggressor is screened due to user Exclusion
I - aggressor has Infinite arrival with respect to the victim
L - aggressor is screened due to Logical correlation
N - aggressor does Not overlap for the worst case alignment
S - aggressor is screened for Small bumps
U - aggressor/victim RC calculation is skipped.
X - aggressor is screened due to aggressor eXclusion
```

Active aggressor is an aggressor that contributes to the worst case alignment scenario. It is reported as 'A'.

PrimeTime analyzes the composite effect of a group of weak aggressors (including filtered ones) for better quality of SI analysis. These aggressors are reported as 'C'.

You can exclude an aggressor from the analysis by using the **case_analysis**, **set_si_delay_analysis -exclude** or **set_coupling_separation** command. These aggressors are reported with attribute 'E'.

An aggressor has infinite window when it has no fixed timing relationship with the victim, for example, an aggressor clock is asynchronous to the victim clock. Also, if the victim or aggressor has the **set_si_delay_analysis -ignore_arrival** command, the aggressor gets the infinite window attribute 'I'.

When PrimeTime SI finds that the victim and aggressors cannot switch together due to the logical relationship (for example, the input and output net of a inverter, they cannot switch in the same direction at the same time), PrimeTime SI chooses the combination of aggressors that can switch together logically and set them to active. The aggressors that cannot switch due to logical relationship are screened with attribute 'L'. The logical relationship could be identified only for buffer/inverter chains.

Due to multiple aggressors and their possible multiple arrival windows there could be multiple alignments which could cause crosstalk delta. PrimeTime SI identifies and reports the crosstalk delta for only the worst case alignment. If the aggressor does not contribute to the worst case alignment scenario, it is screened, and the attribute is set to 'N'.

The switching bumps that are smaller compared to the electrical filtering criteria, are screened. The attribute of these aggressors are set to 'S'. When composite aggressor feature is applied, the SI effect of screened aggressors are considered, and 'S' is replaced by 'C'.

If the aggressor or victim RC calculation is skipped, the attribute is set to 'U'. The common cause of the RC calculation skipping on a net is there is no cell arc driving the net(victim or aggressor) or the driving model could not be created.

The **set_si_aggressor_exclusion** command can set aggressor exclusion relationship among aggressors (for example, the specified aggressors cannot switch in the same direction at the same time), PrimeTime SI chooses the combination of aggressors that can switch together based on alignment and sets them to active. The aggressors that cannot switch due to aggressor exclusion relationship are screened with attribute 'X'.

EXAMPLES

The following example shows the output on a cell arc connected to an RC network. RC delay calculation is performed.

```
pt_shell> report_delay_calculation -thresholds -from U3/CK -to U3/Q

****************************************
Report : delay_calculation
 ...
****************************************

From pin: U3/CK
To pin:   U3/Q
Main Library Units:  1ns  0.001pF  1000kOhm

Library: 'slow'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'SDFFX2'
arc sense: rising_edge
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage  Temp.
   ---------------------------------------------------------------------
      from-pin  50     10->90    50     90->10    1.000     1.080    -40.0
        to-pin  50     10->90    50     90->10    1.000     1.080    -40.0

   RC network on pin 'U3/Q' :
   -------------------------------------------------------
   Number of elements = 2 Capacitances + 1 Resistances
   Total capacitance  = 0.094381 pF
   Total capacitance  = 94.380999 (in library unit)
   Total resistance   = 5.000000 Kohm

                             Rise        Fall
   ------------------------------------------------------------------
   Input transition time  = 0.000000    0.000000  (in library unit)
   Effective capacitance  = 0.069497    0.061939  (in pF)
   Effective capacitance  = 69.496963    61.939442  (in library unit)
   Drive resistance       = 2.020383    1.252620  (in Kohm)
   Output transition time = 0.656019    0.389840  (in library unit)
   Cell delay             = 0.517020    0.356812  (in library unit)
```

The following example shows the output on a net arc. Wire load delay calculation is performed.

```
pt_shell> report_delay_calculation -min -thresholds -from U3/Q  -to U4/A

****************************************
Report : delay_calculation
         -min
 ...
****************************************

From pin: U3/Q
To pin:   U4/A
Main Library Units:  1ns  0.001pF  1000kOhm

arc sense: unate
arc type:  net
```

```
   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage  Temp.
   ---------------------------------------------------------------------------
      from-pin  50     10->90    50     90->10    1.000     1.080    -40.0
        to-pin  50     10->90    50     90->10    1.000     1.080    -40.0

RC delay calculation is being skipped because RC delay calculation was not used for the driver arcs.

Balanced case tree
RC delay: (r_wire/load_count) * (c_pin + c_wire/load_count)
rise:     (0.005 / 1) * (4.381 + (90 / 1))
fall:     (0.005 / 1) * (4.381 + (90 / 1))


Rise delay = 0.471905
Fall delay = 0.471905
```

The following example shows the output on a cell arc. Table lookup into cell the library tables is performed.

```
pt_shell> report_delay_calculation -thresholds -from U1/A -to U1/Y

****************************************
Report : delay_calculation
 ...
****************************************

From pin: U1/A
To pin:   U1/Y
Main Library Units:  1ns  0.001pF  1000kOhm

Library: 'slow'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'INVX2'
arc sense: negative_unate
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage  Temp.
   ---------------------------------------------------------------------------
      from-pin  50     10->90    50     90->10    1.000     1.080    -40.0
        to-pin  50     10->90    50     90->10    1.000     1.080    -40.0

RC delay calculation is being skipped because the driver from-pin is disabled.

Units:  1ns  1pF  1kOhm


 Rise Delay

   cell delay = 0.334919
     Table is indexed by
      (X) input_pin_transition = 0
      (Y) output_net_total_cap = 0.094381
     Relevant portion of lookup table:
                    (X)  0.0420      (X)  0.0660
  (Y)  0.0844       (Z)  0.3141      (Z)  0.3215
  (Y)  0.1706       (Z)  0.6050      (Z)  0.6125

     Z = A + B*X + C*Y + D*X*Y
     A =  0.0168           B =  0.3052
     C =  3.3703           D =  0.0362

  Z = 0.334919
```

```
    scaling result for operating conditions
    multiplying by 1 gives 0.334919

   Fall Delay

     cell delay = 0.215786
       Table is indexed by
        (X) input_pin_transition = 0
        (Y) output_net_total_cap = 0.094381
       Relevant portion of lookup table:
                      (X)  0.0420      (X)  0.0660
    (Y)  0.0844      (Z)  0.2071      (Z)  0.2145
    (Y)  0.1706      (Z)  0.3939      (Z)  0.4013

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0115           B =  0.3079
       C =  2.1650           D =  0.0082

    Z = 0.215786
    scaling result for operating conditions
    multiplying by 1 gives 0.215786

    Cell Delay
    rise:  0.334919
    fall:  0.215786

  Rise delay = 0.334919
  Fall delay = 0.215786

  Units:  1ns  1pF  1kOhm

    Transition rise
     transition = 0.610174
       Table is indexed by
        (X) input_pin_transition = 0
        (Y) output_net_total_cap = 0.094381
       Relevant portion of lookup table:
                      (X)  0.0420      (X)  0.0660
    (Y)  0.0844      (Z)  0.5478      (Z)  0.5478
    (Y)  0.1706      (Z)  1.0854      (Z)  1.0854

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0220           B =  0.0005
       C =  6.2318           D = -0.0058

    Z = 0.610174
    scaling result for operating conditions
    multiplying by 1 gives 0.610174


    Transition fall
     transition = 0.367266
       Table is indexed by
        (X) input_pin_transition = 0
        (Y) output_net_total_cap = 0.094381
       Relevant portion of lookup table:
                      (X)  0.0420      (X)  0.0660
    (Y)  0.0844      (Z)  0.3297      (Z)  0.3297
    (Y)  0.1706      (Z)  0.6536      (Z)  0.6536

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0129           B = -0.0003
       C =  3.7542           D =  0.0029
```

```
  Z = 0.367266
  scaling result for operating conditions
  multiplying by 1 gives 0.367266


Rise transition = 0.610174
Fall transition = 0.367266
```

The following example shows the output on a net arc whose from-pin is a hierarchical pin. The delay of the net is assigned to the net arc whose from-pin is a leaf pin.

```
pt_shell> report_delay_calculation -from U1/clock_out -to U102/CP

****************************************
Report : delay_calculation
 ...
****************************************

RC delay calculation is being skipped because the load to-pin is not an input pin.

From pin: U1/clock_out
To pin:   U102/CP
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net

Net arc from a hierarchical pin.
Arc rise delay = 0 (assigned)
Arc fall delay = 0 (assigned)
To_pin rise transition time = 49.6413 (copied from from_pin)
To_pin fall transition time = 35.2787 (copied from from_pin)
```

The following example shows the report for cell arc when the advanced mode of delay calculation is used. Advanced mode for driver model can be set using the **rc_driver_model_mode** variable. No additional option of **report_delay_calculation** is necessary to report details of the advanced mode of delay calculation.

```
pt_shell> report_delay_calculation -min -thresholds -from sig_in0  -to sig_in0  -nosplit

****************************************
Report : delay_calculation
        -min
 ...
****************************************

From port: sig_in0
To port:   sig_in0
Main Library Units:  1ns  1pF  1kOhm

Library: 'ports'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'sig_in0'
arc sense: positive_unate
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage  Temp.
   ---------------------------------------------------------------------
      from-pin  50     30->70    50     70->30    0.400     1.100    125.0
        to-pin  50     30->70    50     70->30    0.400     1.100    125.0

   RC network on pin 'sig_in0' :
   -----------------------------------------------------
   Number of elements = 8 Capacitances + 7 Resistances
```

```
Total capacitance  = 0.003062 pF
Total capacitance  = 0.003062 (in library unit)
Total resistance   = 0.017983 Kohm

Advanced driver-modeling used for rise and fall.

                           Rise        Fall
-------------------------------------------------------------------
Input transition time  =  0.100000    0.100000  (in library unit)
Effective capacitance  =  0.002625    0.002800  (in pF)
Effective capacitance  =  0.002625    0.002800  (in library unit)
Output transition time =  0.060388    0.047040  (in library unit)
Cell delay             =  0.050937    0.045125  (in library unit)
```

The following example shows the report for net arc when the advanced mode of delay calculation is used. You can set the advanced mode for the receiver model by using the **rc_receiver_model_mode** variable. No additional option of the **report_delay_calculation** command is necessary to report details of the advanced mode of delay calculation. In addition to the existing information, the pin capacitances of the advanced receiver model and the transition values used to calculate the pin capacitances of the advanced receiver model are also reported.

```
pt_shell> report_delay_calculation -max -thresholds -from sig_in0  -to cell0/A  -nosplit

From port: sig_in0
To pin:   cell0/A
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage  Temp.
   ---------------------------------------------------------------------
      from-pin  50     30->70    50     70->30    0.400     1.100    125.0
        to-pin  50     30->70    50     70->30    0.400     1.100    125.0

   RC network on pin 'sig_in0' :
   -------------------------------------------------------
   Number of elements = 8 Capacitances + 7 Resistances
   Total capacitance  = 0.003062 pF
   Total capacitance  = 0.003062 (in library unit)
   Total resistance   = 0.017983 Kohm

   Advanced receiver-modeling used for rise and fall.

   Advanced Receiver Model
                              Rise            Fall
   -------------------------------------------------------------------
   Receiver model capacitance  1 = 0.001966    0.001888  (in library unit)
   Receiver model capacitance  2 = 0.002328    0.002160  (in library unit)
   Receiver model transition   1 = 0.059192    0.037666  (in library unit)
   Receiver model transition   2 = 0.059192    0.037666  (in library unit)

                               Rise        Fall
   -------------------------------------------------------------------
   Net delay                =   0.000024    0.000064  (in library unit)
   Transition time          =   0.059192    0.037599  (in library unit)
   From_pin transition time =   0.059078    0.037666  (in library unit)
   To_pin transition time   =   0.059192    0.037599  (in library unit)
   Net slew degradation     =   0.000114   -0.000067  (in library unit)
```

The **define_scaling_lib_group** command allows you to define a group of libraries that PrimeTime interpolates between for voltage or temperature scaling. The following example shows the output on a net arc when scaling of the receiver model is done. The report indicates that scaling was done and also reports all the scaling libraries

that were used for scaling the model.

```
pt_shell > report_delay_calculation -from I1/Z -to I2/B

From pin: I1/Z
To pin:   I2/B
Main Library Units:  1ns  0.001pF  1000kOhm

arc sense: unate
arc type:  net

   RC network on pin 'I1/Z' :
   -------------------------------------------------------
   Number of elements = 2 Capacitances + 1 Resistances
   Total capacitance  = 0.815687 pF
   Total capacitance  = 815.686687 (in library unit)
   Total resistance   = 8.323139 Kohm

   Scaling library pin group used for rise and fall.
   Scaling libraries used for receiver model : tc300c_0.85  tc300c_1.05

                              Rise        Fall
   ---------------------------------------------------------------
   Net delay                =   2.406542    2.836394  (in library unit)
   Transition time          =   5.312853    4.983530  (in library unit)
   From_pin transition time =   0.804747    0.328441  (in library unit)
   To_pin transition time   =   5.312853    4.983530  (in library unit)
   Net slew degradation     =   4.508106    4.655089  (in library unit)
```

The following run shows the output on a net arc connected to a port. RC delay calculation with crosstalk is performed.

```
pt_shell> report_delay_calculation  -crosstalk -from u2/Z -to o2

****************************************
Report : delay_calculation
Design : xtalk_test
****************************************

From pin: u2/Z
To port:   o2
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net
Annotated max rise net delta delay: 0 arc delay: 0.0151697
Annotated max fall net delta delay: 0 arc delay: 0.0153037

   RC network on pin 'u2/Z' :
   -------------------------------------------------------
   Number of elements = 5 Capacitances + 4 Resistances
   Total capacitance  = 0.300000 pF
   Total capacitance  = 0.300000 (in library unit)
   Total resistance   = 0.100000 Kohm

                              Rise        Fall
   ---------------------------------------------------------------
   Net delay                = 0.015170    0.015304  (in library unit)
   Transition time          = 0.940437    0.542964  (in library unit)
   From_pin transition time = 0.940063    0.542372  (in library unit)
   To_pin transition time   = 0.940437    0.542964  (in library unit)
   Net slew degradation     = 0.000374    0.000592  (in library unit)

Annotated max rise delta transition: 0 pin transition: 0.940437
```

```
Annotated max fall delta transition: 0 pin transition: 0.542964
Reporting for Crosstalk:
Victim net name:                                   o2
Number of aggressors:                              1
Number of effective (non-filtered) aggressors:     1
Net is reselected:                                 true
Victim driver rail voltage(VDD):                   2.700000
si_xtalk_delay_analysis_mode:                      all_paths
si_analysis_logical_correlation_mode:              true
Crosstalk composite aggressor mode:                disabled

Attributes:
  A - aggressor is Active
  C - aggressor is a composite aggressor
  E - aggressor is screened due to user Exclusion
  I - aggressor has Infinite arrival with respect to the victim
  L - aggressor is screened due to Logical correlation
  N - aggressor does Not overlap for the worst case alignment
  S - aggressor is screened for Small bumps
  U - aggressor/victim RC calculation is skipped
  X - aggressor is screened due to aggressor eXclusion

Victim is rising:
  Victim           Coupling    Driver        Clocks
   Net             Cap         Lib Cell
  -----------      ---------   ----------    ---------
  o2               0.200000    BF1T4_D        CLK2

  Aggressor        Coupling    Driver        Clocks        Attributes    Switching Bump
   Net             Cap         Lib Cell                                  (ratio of VDD)
 -------------     ---------   ---------     ---------     ------------  ----------------
  o1               0.200000    BF1T4_D        CLK1          N               -

Victim is falling:
  Victim           Coupling    Driver        Clocks
   Net             Cap         Lib Cell
  -----------      ---------   ----------    ---------
  o2               0.200000    BF1T4_D        CLK2

  Aggressor        Coupling    Driver        Clocks        Attributes    Switching Bump
   Net             Cap         Lib Cell                                  (ratio of VDD)
  -----------      ---------   ---------     ---------     ------------  ----------------
  o1               0.200000    BF1T4_D        CLK1          N               -
```

SEE ALSO

```
define_scaling_lib_group(2)
get_timing_arcs(2)
report_lib(2)
report_lib_groups(2)
report_timing(2)
set_si_aggressor_exclusion(2)
rc_driver_model_mode(3)
rc_receiver_model_mode(3)
si_analysis_logical_correlation_mode(3)
si_xtalk_composite_aggr_mode(3)
si_xtalk_delay_analysis_mode(3)
```

report_design

Shows information about the current design.

SYNTAX

```
status report_design
   [-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting in the report when a field exceeds the column width.

DESCRIPTION

This command shows the following information about the current design:

- Operating conditions
- Wire load
- Design rules
- Timing ranges

EXAMPLES

The following example shows a design report.

```
pt_shell> report_design

****************************************
Report : design
Design : counter
 ...
****************************************

Design Attribute                 Value
--------------------------------------------------------------------------------
```

```
Operating Conditions:
  operating_condition_max_name   WCCOM
  process_max                    1.50
  temperature_max                70.00
  voltage_max                    4.75
  tree_type_max                  worst_case_tree

Wire Load:                       (use report_wire_load for more information)
  wire_load_mode                 top
  wire_load_model_max            --
  wire_load_selection_group_max  --
  wire_load_min_block_size       0

Design Rules:
  max_capacitance                0.5
  min_capacitance                --
  max_fanout                     5.6
  min_fanout                     --
  max_transition                 0.8
  min_transition                 --
  max_area                       --

Timing Ranges:
  fastest_factor                 --
  slowest_factor                 --
```

SEE ALSO

report_clock(2)
report_port(2)

report_design_mismatch

Reports all design mismatches found during linking.

SYNTAX

```
string report_design_mismatch
   [-class cell | net | pin]
```

ARGUMENTS

-class cell | net | pin

For any pin that is found to have mismatches (such as instance and reference have different directions on the pin), only the associated objects (cells, nets, pins) that belong to this class are reported. The possible values are "cell", "net" and "pin".

DESCRIPTION

When the **link_allow_design_mismatch** variable is **true**, linking succeeds even when there are mismatches found. This allows you to gather as much useful information as possible even in the early stages of design. This command provides a report of mismatches for the current design.

Common causes of those mismatches include the following (listed in order of increasing priority):

1. A pin has different directions in instance and reference.
2. A pin of instance does not exist in reference.
3. A bus has different widths in instance and reference.

If a cell, pin, or net has a mismatch issue, the **is_design_mismatch** attribute returns **true** for the affected cell, pin, or net object. If the pin with a mismatch is on a bus, all other bits of the bus are also marked with the **is_design_mismatch** attribute.

Note that if a cell or net has more than one mismatched issue, only the one with the highest priority is reported.

EXAMPLES

The following report example shows that the u1/Z and u2/Z pins have direction mismatches between instance and

reference. The b net is connected to both u1/Z and u2/Z.

```
pt_shell> report_design_mismatch

****************************************
Report : report_design_mismatch
 ...
****************************************

pin               mismatch type
----------------------------------------
u1/Z              Pin direction mismatch
u2/Z              Pin direction mismatch
----------------------------------------
cell              mismatch type
----------------------------------------
u1                Pin direction mismatch
u2                Pin direction mismatch
----------------------------------------
net               mismatch type
----------------------------------------
b                 Pin direction mismatch
```

SEE ALSO

link_design(2)
link_allow_design_mismatch(3)

report_disable_timing

Reports disabled timing arcs in the current design.

SYNTAX

```
string report_disable_timing
   [-nosplit]
   [cells_or_ports]
```

Data Types

cells_or_ports	collection

ARGUMENTS

-nosplit

Prevents line splitting to facilitate writing software to extract information from the report output. If you do not use this option, the report is shown in fixed-width columns.

cells_or_ports

Limits disabled arc reporting to the specified list of cells or ports. Provides the list or collection of cells or ports as an argument to the command.

DESCRIPTION

This command reports disabled timing arcs in the current design. Timing arcs can be disabled in several ways. You can disable a timing arc by using the **set_disable_timing** command. The following symbols are used to explain why a timing arc is disabled:

```
c : The propagation of case-analysis values

C : The presence of conditional arcs (arcs that have a when statement defined in the library)

d : The presence of default arcs (when the timing_disable_cond_default_arcs variable
is set to true)

f : Disabling of false net-arcs

l : Loop breaking

L : db inherited loop breaking
```

```
m : Mode

p : The propagation of constant values

u : Arcs disabled by using the set_disable_timing command on cell arcs.

U : Arcs disabled by using the set_disable_timing command on library cell arcs.
```

EXAMPLES

The following example shows that the timing arc of the n2_i cell from pin A to pin Z is disabled.

```
pt_shell> set_disable_timing {n2_i} -from A -to Z
pt_shell> report_disable_timing

****************************************
Report : disable_timing
 ...
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs

Cell or Port         From    To      Sense           Flag  Reason
-------------------------------------------------------------------
n2_i                 A       Z       negative_unate  u

1
```

The following example specifies that the timing arc of the ff4 cell from pin CP to pin TE and TI are disabled as a result of a case-analysis constant of 0 propagated to pin TE of cell ff4. Note that the actual case value is set on another pin A and the propagation path to ff4/TE is inverting.

```
pt_shell> set_case_analysis 0 {p_in in0}
pt_shell> report_disable_timing

****************************************
Report : disable_timing
Design : middle
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs
```

```
Cell or Port    From    To    Sense                  Flag  Reason
-----------------------------------------------------------------------
o_reg4          CP      D     hold_clk_rise          c     D = 0
o_reg4          CP      D     setup_clk_rise         c     D = 0
o_reg4          CP      CD    recovery_rise_clk_rise c     D = 0
p_out                                                c     p_out = 1

1
```

To view disabled arcs in specific cells, provide a list or collection of the required cells as an argument to the command. To view disabled arcs for the o_reg4 instance, do the following:

```
pt_shell> report_disable_timing  [get_cells { o_reg4 }]

****************************************
Report : disable_timing
Design : middle
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs

Cell or Port   From   To   Sense                 Flag    Reason
-----------------------------------------------------------------
o_reg4         CP     D    hold_clk_rise           c     D = 0
o_reg4         CP     D    setup_clk_rise          c     D = 0
o_reg4         CP     CD   recovery_rise_clk_rise  c     D = 0

1
```

SEE ALSO

```
remove_disable_timing(2)
set_disable_timing(2)
```

report_driver_model

Displays the driver model for a library cell timing arc used to drive annotated parasitics.

SYNTAX

```
integer report_driver_model
   -lib_cell lib_cell
   -from_pin from_pin
   -to_pin to_pin
   -rise_slew rise_slew
   -fall_slew fall_slew
   -capacitance capacitance
```

Data Types

lib_cell	string
from_pin	string
to_pin	string
rise_slew	float
fall_slew	float
capacitance	double

ARGUMENTS

-lib_cell *lib_cell*

Specifies the name of the library cell for which the driver model is computed. Specify the name in the *library_name/cell_name* format.

-from_pin *from_pin*

Specifies the startpoint of a timing arc through the *lib_cell* value.

-to_pin *to_pin*

Specifies the endpoint of a timing arc through the *lib_cell* value.

-rise_slew *rise_slew*

Specifies the rise time in library units on the *from_pin* value.

-fall_slew *fall_slew*

Specifies the fall time in library units on the *from_pin* value.

-capacitance *capacitance*

Specifies the load in library units on the *to_pin* value.

DESCRIPTION

The **report_driver_model** command provides both the library data and the resulting driver model parameters for the specified library arc and conditions. This information is useful for validating library data or PrimeTime driver models used in delay calculation with annotated parasitics.

For each timing arc between the specified pins, the delay, slew, and sensitivities (the changes in delay and slew resulting from a small change in load capacitance) are displayed. This information can be compared to simulation results using the same input slew and output load to measure library interpolation errors. If the current design is in min-max mode, data for both operating conditions is displayed.

Slews are displayed without derating (they are shown as the time between the trip-points.).

In addition to the library data, the driver model parameters used in delay calculation with annotated parasitics are displayed. Currently, PrimeTime supports only the simplified driver model (SDM) with three parameters: rd (the drive resistance through which a linear voltage ramp is applied), tz (the ramp start time relative to the arc input threshold time), and dt (the full-swing ramp duration).

The driver model is constructed to match the library delay, slew, and sensitivity for the specified input slew and output load.

Problematic library data can prevent the construction of a driver model altogether. In this case, the driver model parameters displays as zero, and the check shows FAIL. The most common reason for this error is that the library data not indicating an increasing delay or slew for increasing output capacitance; this indicates a nonpositive drive resistance. Another common reason is incorrect trip-point settings.

EXAMPLES

The following report shows the output for the driver model for a library cell timing arc.

```
pt_shell> report_driver_model \
  -lib_cell [get_lib_cell -of_objects I0] \
  -from_pin A -to_pin Y \
  -rise_slew 0.0384 -fall_slew 0.0384 -capacitance 0.115194


****************************************
Report : driver_model
Driver : core/inv27:A-->Y
Version: 2000.11
Date   : Thu Aug  7 14:43:08 2000
****************************************

Accuracy Settings:             Rise  Fall
    slew_lower_threshold     =  30%   30%
    slew_upper_threshold     =  70%   70%
    input_threshold          =  50%   50%
    output_threshold         =  50%   50%
    slew_derate_from_library =       1

Library Inputs:  (in library units)
    Rising  input slew      =  0.038400
    Falling input slew      =  0.038400
```

```
    Output load capacitance =  0.115194

  Driver model for sense 'negative-unate':
    (max)        Rise      Fall
    ---------------------------------------------------
    load    =  0.115194  0.115194  (pF)
    delay   =  0.071811  0.044181  (ns)
    slew    =  0.043008  0.026270  (ns without derate)
    d-load  =  0.001152  0.001152  (pF)
    d-delay =  0.000392  0.000264  (ns)
    d-slew  =  0.000332  0.000160  (ns without derate)
    rd      =  0.406621  0.249222  (kohm)
    tz      = -0.000323  0.000139  (ns)
    dt      =  0.070798  0.043157  (ns)
    error   =  0.853807  0.950170  (%)
    check   =    pass      pass
```

SEE ALSO

```
report_delay_calculation(2)
rc_driver_model_mode(3)
```

report_em_options

Reports configuration information for electromigration analysis.

SYNTAX

string **report_em_options**

ARGUMENTS

No arguments are needed

DESCRIPTION

The command generates a report showing configuration options for electromigration analysis, including temperature, snap temperature, EM rule scale factor, absolute minimum width limit, and relative minimum width limit. The electromigration analysis configuration information can be specified by using **set_em_options** command.

EXAMPLES

The following example specifies and reports configuration options for electromigration violation checking:

```
pr_shell> set_em_options -temperature 125 -snap_temperature_to_rule 100 -scale_factor 0.8
1
pr_shell> report_em_options
****************************************
Report : Options for EM checking
Design : design
Version: I-2013.12-DEV
Date   : Thu May 23 15:32:23 2013
****************************************

Temperature:   125.00 C

Snap Temperature to Rule: ON

Scale Factor:    8.00e-01

Width Limit: NONE
```

```
Relative Width Limit:    80.00 %

1
```

SEE ALSO

```
set_em_options(2)
report_em_rules(2)
report_em_violations(2)
update_em(2)
```

report_em_rules

Reports the electromigration rules currently loaded in the session.

SYNTAX

```
string report_em_rules
```

ARGUMENTS

None

DESCRIPTION

The command generates a report showing the electromigration (EM) rules that are currently loaded into the session. The electromigration rules can come from two sources: rules stored within Milkyway (either from the technology file or Milkyway cached EM rules), or rules that have been read in as an ALF file by **read_em_rules**.

Electromigration rules establish layer-by-layer current density limits for the layout geometries. The rules can establish limits on the average current, peak current, and RMS current. Electromigration rules are typically temperature and width dependent, but may also establish other dependencies, such as route length.

EXAMPLES

The following example shows how to generate an electromigration rule report:

```
pr_shell> report_em_rules


****************************************
Report : EM rules
Design : design
Version: I-2013.12-DEV
Date   : Thu May 23 15:41:43 2013
****************************************

Source: Milkyway
Units:
  WIDTH  : 1e-06 Meter
  AREA   : 1e-12 Meter**2
  CURRENT: 0.001 Amp
```

```
Layer: metal1
Mode : average
              Temperature
             +-----------+-----------+-----------+-----------+
    Width    |        -40|         25|        105|        125|
  -----------+-----------+-----------+-----------+-----------+
        0.100|      0.198|      1.980|      0.198|      1.980|
        1.000|      0.198|      1.980|      0.198|      1.980|
  -----------+-----------+-----------+-----------+-----------+
                          (I,J)=Max_Current

Layer: metal2
Mode : average
              Temperature
             +-----------+-----------+-----------+-----------+
    Width    |        -40|         25|        105|        125|
  -----------+-----------+-----------+-----------+-----------+
        0.100|      0.254|      2.540|      0.254|      2.540|
        1.000|      0.254|      2.540|      0.254|      2.540|
  -----------+-----------+-----------+-----------+-----------+
                          (I,J)=Max_Current

Layer: via1
Mode : average
  Temperature| Max_Current
  -----------+-------------
          -40|       0.060
           25|       0.060
          105|       0.060
          125|       0.060

Layer: via2
Mode : average
  Temperature| Max_Current
  -----------+-------------
          -40|       0.080
           25|       0.080
          105|       0.080
          125|       0.080
```

SEE ALSO

calculate_rail_voltage(2)
read_em_rules(2)
report_em_violations(2)

report_em_violations

Reports the electromigration rule violations based on the current rail analysis result.

SYNTAX

```
string report_em_violations
   [supply_nets]
   [-scale_factor scale_value]
   [-layer layer_list]
   [-temperature temperature_value]
   [-width_limit width_value]
   [-relative_width_limit percentage_value]
   [-advanced_em]
   [-error_cell error_cell_name]
```

Data Types

```
list supply_nets
float scale_value
list layer_list
float width_value
float temperature_value
float percentage_value
string error_cell_name
```

ARGUMENTS

supply_nets

Specifies a list of supply nets to be checked for electromigration violations. Items within the list can be supply net name patterns or collections. The default is to report violations on all of the supply nets in the current rail analysis result.

-scale_factor *scale_value*

Specifies a uniform scale factor to apply to the current threshold values in the electromigration rules before testing for electromigration violations.

By default, the *scale_value* is the value set by the **set_em_options** command.

-layer *layer_list*

Specifies the layers to be included in the electromigration violation report. The layers can be specified as Milkyway layer names or Milkyway layer ID numbers. Even if the polygons carry current that violates an electromigration rule, the tool does report these polygons if the polygons are not on either of the specified layers.

By default, all layers are included in the electromigration violation report.

-width_limit ***width_value***

Specifies an absolute minimum width, in microns, for polygons to be included in the electromigration violation report. If the width of a polygon carrying current that violates an electromigration rule is smaller than the *width_value*, it is not included in the electromigration violation report.

By default, the *width_value* is the value set by the **set_em_options** command.

-temperature ***temperature_value***

Specifies a global temperature for electromigration analysis, in Celcius.

-relative_width_limit ***percentage_value***

Specifies a minimum width, as a percentage of the layer minimum width defined in the technology file, for polygons to be included in the electromigration violation report. If the width of a polygon carrying current that violates an electromigration rule is smaller than the defined percent of the minimum width for its layer, it is not included in the electromigration violation report.

By default, the percentage_value is the value set by the **set_em_options** command.

-advanced_em

Enables the advanced electromigration analysis using the advanced electromigration rules loaded with the **read_em_rules** command.

-error_cell ***error_cell_name***

Creates a Milkyway error cell to be viewed in the IC Compiler error browser.

DESCRIPTION

The command performs electromigration analysis and reports the rule violations for the nets included in the current rail result. This command uses the per-shape electrical currents calculated as a side effect of rail analysis. You must run the command after the **calculate_rail_voltage** command is run or after a rail result that contains the voltage drop analysis result has been set and loaded by the **current_rail_result** command.

Electromigration analysis is used to determine whether the current density at a location in the circuit exceeds the defined electromigration rule limits. The current density limits are intended to control the risk that current-carrying geometries will not be degraded by excessive current flow. High current densities damage the wires and damaged current-carrying wires can ultimately lead to shorts and opens within the metal, causing circuit failure. The silicon device foundries set the current density limit rules of their related metallurgy to ensure reliability. See the **read_em_rules** man page for more details.

You can use the *-scale_factor* option to assign a uniform scaling factor to the defined current density limits before performing the violation check. Oftentimes this is done when you want to set an additional margin to the limits. For example, if you want to verify that your design does not have current densities that exceed more than 90% of the allowed value, set the scale factor to 0.9. Use the **report_em_rules** command to see the currently defined electromigration rules.

Use the *-temperature* option to specify a temperature when evaluating temperature dependent electromigration constraints. By default, the command uses the temperature of the default operating condition in the first library in the link path.

You can use the *-error_cell* option to create a Milkyway error cell to be viewed in the IC Compiler error browser.

Note that the current direction (CUR_DIR) column has six possible values: Left and Right for horizontal wires, Up and Down for vertical wires, and Up and Down for vias.

EXAMPLES

The following example performs electromigration analysis on the VDD net after rail analysis is complete, using the current rules:

```
pr_shell> calculate_rail_voltage VDD
pr_shell> report_em_violations

****************************************
Report : EM violations
Design : test
Version: D-2010.06-Alpha2
Date   : Fri Apr  2 10:12:34 2010
****************************************

Scale Factor : 1.000
Temperature  : 125 C
Units:
  WIDTH      : 1e-06 Meter
  LENGTH     : 1e-06 Meter
  AREA       : 1e-12 Meter**2
  CURRENT    : 0.001 Amp
Keys:
  L_NAME     : layer name
  L_NUM      : layer number
  THRESHOLD  : EM threshold for current
  W|A        : width or area (rule dependent)
  REC W|A    : recommended width or area (rule dependent) needed to resolve violation
  L          : length (rule dependent)

L_NAME L_NUM MODE    CURRENT (>THRESHOLD) W|A    REC W|A (LEFT, BOTTOM)     (RIGHT, TOP)         L
-------------------------------------------------------------------------------------------------------
VIA8   58    average 0.064   (>0.049)     2.722  3.580   (249.370, 479.920) (250.630, 482.080)      N
VIA8   58    average 0.065   (>0.049)     2.722  3.640   (249.370, 419.920) (250.630, 422.080)      N
VIA8   58    average 0.053   (>0.049)     2.722  2.946   (249.370, 359.920) (250.630, 362.080)      N
VIA7   57    average 0.064   (>0.049)     2.722  3.580   (249.370, 479.920) (250.630, 482.080)      N
VIA7   57    average 0.065   (>0.049)     2.722  3.640   (249.370, 419.920) (250.630, 422.080)      N
VIA7   57    average 0.053   (>0.049)     2.722  2.946   (249.370, 359.920) (250.630, 362.080)      N
1
```

The following example loads a new set of electromigration rules after rail analysis has completed and then performs an electromigration analysis with an additional 30% of margin:

```
pr_shell> calculate_rail_voltage VDD VDD1
pr_shell> read_em_rules -format alf NewRules.alf
pr_shell> report_em_violations -scale_factor 0.7

****************************************
Report : EM violations
Design : Test
Version: D-2010.06-Alpha2
Date   : Fri Apr  2 10:18:46 2010
****************************************

Scale Factor : 0.700
Temperature  : 125 C
Units:
```

```
  WIDTH      : 1e-06 Meter
  LENGTH     : 1e-06 Meter
  AREA       : 1e-12 Meter**2
  CURRENT    : 0.001 Amp
Keys:
  L_NAME     : layer name
  L_NUM      : layer number
  THRESHOLD  : EM threshold for current
  W|A        : width or area (rule dependent)
  REC W|A    : recommended width or area (rule dependent) needed to resolve violation
  L          : length (rule dependent)

L_NAME L_NUM MODE    CURRENT (>THRESHOLD) W|A    REC W|A (LEFT, BOTTOM)     (RIGHT, TOP)         L
-------------------------------------------------------------------------------------------------------
VIA8   58    average 0.064   (>0.034)     2.722  5.115   (249.370, 479.920) (250.630, 482.080)      N
VIA8   58    average 0.065   (>0.034)     2.722  5.200   (249.370, 419.920) (250.630, 422.080)      N
VIA8   58    average 0.053   (>0.034)     2.722  4.209   (249.370, 359.920) (250.630, 362.080)      N
VIA8   58    average 0.037   (>0.034)     2.722  2.950   (249.370, 299.920) (250.630, 302.080)      N
VIA7   57    average 0.064   (>0.034)     2.722  5.115   (249.370, 479.920) (250.630, 482.080)      N
VIA7   57    average 0.065   (>0.034)     2.722  5.200   (249.370, 419.920) (250.630, 422.080)      N
VIA7   57    average 0.053   (>0.034)     2.722  4.209   (249.370, 359.920) (250.630, 362.080)      N
VIA7   57    average 0.037   (>0.034)     2.722  2.950   (249.370, 299.920) (250.630, 302.080)      N
1
```

The following example loads a new set of electromigration rules in TLUPLUS format after rail analysis has completed and then performs an electromigration analysis at 110 degree temperature:

```
pr_shell> calculate_rail_voltage VDD
pr_shell> read_em_rules -format tluplus NewRules.tluplus -design2itf_map layer_mapping
pr_shell> report_em_violations -temperature 110

Information: use relative width limit, value is 80.00 %.
Information: temperature 110C from the nominal operating condition in the first library in the link p
****************************************
Report : EM violations
Design : TEST
Version: J-2014.06-alpha
Date   : Mon Apr 14 17:25:42 2014
****************************************

----------------------------------------
Section: EM Violations
----------------------------------------
Scale Factor : 1.000
Temperature  : 110 C
Units:
  WIDTH      : 1e-06 Meter
  LENGTH     : 1e-06 Meter
  AREA       : 1e-12 Meter**2
  CURRENT    : 0.001 Amp
Keys:
  L_NAME     : layer name
  L_NUM      : layer number
  THRESHOLD  : EM threshold for current
  W|A        : width or area (rule dependent)
  REC W|A    : recommended width or area (rule dependent) needed to resolve violation
  L          : length (rule dependent)

NET: VDD
L_NAME L_NUM MODE    CURRENT   (>THRESHOLD) W|A       REC W|A   (LEFT, BOTTOM)      (RIGHT, TOP)
-------------------------------------------------------------------------------------------------------
Unknown layer: fpoly_svt
metal2    32 average     0.168>       0.155     0.050        NA(  31.704    4.583) (  31.896    4.633
```

```
metal4    34 average     0.168>       0.155     0.050        NA(  31.704    4.583) (  31.896    4.633
...
```

SEE ALSO

calculate_rail_voltage(2)
current_rail_result(2)
read_em_rules(2)
report_em_rules(2)
set_em_options(2)
report_em_options(2)

report_etm_arc

Reports the data and clock paths traversed while extracting a particular timing arc.

SYNTAX

```
string report_etm_arc
   [-from from_object]
   [-rise_from rise_from_object]
   [-fall_from fall_from_object]
   [-to to_object]
   [-rise_to rise_to_object]
   [-fall_to fall_to_object]
   [-arc_type arc_type]
   [-include path_type_list]
   [-context_borrow]
   [-latch_level levels]
   [-library_cell]
   [-etm_report er_file_name]
   [-netlist_report nr_file_name]
   [-significant_digits digits]
```

Data Types

from_object	list
rise_from_object	list
fall_from_object	list
to_object	list
rise_to_object	list
fall_to_object	list
arc_type	list
path_type_list	list
levels	integer
er_file_name	string
nr_file_name	string
digits	integer

ARGUMENTS

-from *from_object*

Specifies a port or a clock from which the arc of interest originates. The sense at the starting point can be either rising or falling. Substitute one of the following valid values for *from_object*: **-fall_from**, **-from**, or **-rise_from**.

-rise_from *rise_from_object*

Specifies a port or a clock from which the arc of interest originates. The sense at the startpoint must be rising.

-fall_from ***fall_from_object***

Specifies a port or a clock from which the arc of interest originates. The sense at the startpoint must be falling.

-to ***to_object***

Specifies a port or a clock at which the arc of interest terminates. The sense at the endpoint can be either rising or falling. Substitute one of the following valid values for *to_object*: **-fall_to**.,**-rise_to**, or **-to**.

-rise_to ***rise_to_object***

Specifies a port or a clock at which the arc of interest terminates. The sense at the endpoint must be rising.

-fall_to ***fall_to_object***

Specifies a port or a clock at which the arc of interest terminates. The sense at the endpoint must be falling.

-arc_type ***arc_type***

Specifies the type of arc reported between the start and endpoint. Currently, the **report_etm_arc** command can only accept one *arc_type* value. Substitute one of the following valid values for *arc_type*: **clock_gating_hold**, **clock_gating_setup**, **hold**, **max_combo_delay**, **max_seq_delay**, **min_seq_delay**, **min_combo_delay**, **recovery**, **removal**, or **setup**.

-include ***path_type_list***

Specifies whether the clock path or netlist path should also be reported. Substitute one or more of the following valid values for *path_type_list*: **clock_path** and **netlist_path**. Specifying **clock_path** prints a complete clock path in the generated report. Specifying **netlist_path** prints in the report a corresponding netlist path that was used by model validation. Specifying both **clock_path** and **netlist_path** prints a complete ETM Report (consisting of data and clock path) followed by a complete Netlist Report (consisting of data and clock path).

-context_borrow

Specifies that PrimeTime determines the latches on an interface that borrow, based on input port arrival times and clock definitions specified at the time of model extraction. This option is one of three that specify the model extraction environment in which the debugging is done: **-context_borrow**, **-latch_level**, and **-library_cell**. The **-context_borrow** and **-latch_level** options are mutually exclusive.

-latch_level ***levels***

Specifies the number of levels of latch borrowing that occur at the interface of a design. This option is one of three that specify the model extraction environment in which the debugging is done: **-context_borrow**, **-latch_level**, and **-library_cell**. The **-context_borrow** and **-latch_level** options are mutually exclusive.

-library_cell

Specifies that the model generates as a library cell. This option eliminates boundary nets, and the boundary parasitics become part of the model. This option is one of three that specify the model extraction environment in which the debugging is done: **-context_borrow**, **-latch_level**, and **-library_cell**.

-etm_report ***er_file_name***

Specifies that the ETM report must print to a file named er_file_name.

-netlist_report ***nr_file_name***

Specifies that the Netlist report must print to a file named nr_file_name.

-significant_digits ***digits***

Specifies the number of digits after the decimal point is displayed for time values in the generated report.

Substitute one of the following valid values for *digits*: an integer from 0 to 13. The default is 2. You can use this option to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

DESCRIPTION

Reports the data and clock paths traversed while extracting a particular timing arc. It is used to debug a discrepancy between an ETM and a netlist reported by the **write_interface_timing** and **compare_interface_timing** model validation commands.

By default, it reports only the data path traversed by an ETM while extracting a timing arc. You can use the **-include clock_path** option to report the clock path traversed. You can use the **-include netlist_path** option to compare the path traversed by model extraction to the path traversed by the PrimeTime timing engine. For the purpose of comparing ETM paths to netlist paths, using the **-include netlist_path** option is recommended over using the **report_timing** command.

You must use the same model extraction options and environment variables when generating a model as when debugging the model. This ensures that the same arc is extracted during model extraction and debugging. Therefore, the values of environment variables used by the **extract_model** command are also valid for the **report_etm_arc** command. The **-context_borrow**, **-latch_level**, and **-library_cell** options specify how to perform extraction.

EXAMPLES

The following example generates a **report_etm_arc** command from a discrepancy shown by model validation. If the **compare_interface_timing** command yields a failure as:

```
RBUS_RnotW(r)  ADC_SCLK_IN(r)  setup  17.18  18.24  1.06  FAIL

pt_shell> report_etm_arc -rise_from RBUS_RnotW \
            -rise_to [get_clock ADC_SCLK_IN] -arc_type setup \
            -include {clock_path netlist_path}
```

The following example generates a **report_etm_arc** command for a clock to output path. From the following model validation discrepancy:

```
ADC_SCLK_IN(r) ADC_LRCLK_OUT(f)  FALL  2.04  2.71  0.67  FAIL

pt_shell> report_etm_arc \
            -rise_from [get_clock ADC_SCLK_IN] \
            -fall_to ADC_LRCLK_OUT -arc_type min_seq_delay \
            -include {clock_path netlist_path}
```

SEE ALSO

compare_interface_timing(2)
extract_model(2)
report_timing(2)

```
write_interface_timing(2)
```

report_exceptions

Reports timing exceptions.

SYNTAX

```
status report_exceptions
   [-from from_list]
   [-rise_from rise_from_list]
   [-fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list]
   [-rise_to rise_to_list]
   [-fall_to fall_to_list]
   [-ignored]
   [-nosplit]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list

ARGUMENTS

-from *from_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The report includes only paths that start at the objects in the *from_list* value. Using this option limits the report to information that was set using a path-based command (for example, the **set_multicycle_path** command) with the **-from** option. The **-from**, **-rise_from**, and **-fall_from** options are mutually exclusive.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the **-from**, **-rise_from**, and **-fall_from** options.

-through *through_list*

Specifies a list of pins or ports. The report includes only paths that go through the pins or ports on the *through_list* value. Using this option limits the report to information that was set using a path-based command (for example, the **set_multicycle_path** command) with the **-through** option. You can use this option more than one time in the same command.

-rise_through *rise_through_list*

Specifies a list of pins or ports (the same as the **-through** option) except that the paths must have a rising transition at the through points. You can use this option more than one time in the same command.

-fall_through *fall_through_list*

Specifies a list of pins or ports (the same as the **-through** option) except that the paths must have a falling transition at the through points. You can use this option more than one time in the same command.

-to *to_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The report is to include only paths that end at the objects in the *to_list* objects. Using this option limits the report to information that was set using a path-based command (for example, the **set_multicycle_path** command) with the **-to** option. The **-to**, **-rise_to**, and **-fall_to** options are mutually exclusive.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can only use one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the **-to**, **-rise_to**, and **-fall_to** options.

-ignored

Indicates that the report lists path timing exceptions that are set on the current design, but are completely ignored. For example, a false path might be specified from port A to port Z1, but if there is no timing path between those points, the path is ignored. Use the **-from** or **-to** options to limit the report to certain paths. Ignored exceptions on objects are also included in the report; for example, an exception placed by the **set_max_time_borrow** command on a cell other than a level-sensitive latch.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_exceptions** command reports information about timing exceptions for the current design.

PrimeTime accepts a timing exception if it alters the constraints of the path. If path constraints are different in the absence of a timing exception, the exception alters the path constraint.

The **report_exceptions** command attempts to identify reasons an exception is ignored, fully or partially, and classifies these into one of the following: invalid startpoints, invalid endpoints, non-existent paths, or overridden paths. This is reported as flags in the Ignored column. When the **-ignored** option is used, the reported flags indicates why an exception is fully ignored. Otherwise, the reported flags indicates why a subset of the exception specified paths were ignored.

By default (without the **-ignored** option), **report_exceptions** displays dominant exceptions. An exception only appears in this section if a path exists for which this exception applies. That is, a path exists that

- Matches the exception's topological and setup/hold, rise/fall criteria
- Has a valid startpoint and endpoint
- Satisfies no higher-priority exception

If no such path exists, the exception is reported in the ignored report with explanatory reasons why the exception was ignored. Even if the exception is dominant, some parts of its specification might be unsatisfiable. Consequently, each of the following reason flags can appear beside an exception whether the exception is dominant or ignored.

- p: invalid path
  At least one of the pins in through or to sets is not reachable from any pin in its predecessor (-from or -through) pin set, the specified exception is deemed to have an invalid path specification.
- f: invalid start point
  At least one of the pins in from set does not launch a signal. This flag is shown only if a from set is specified in the exception.
- t: invalid endpoint
  At least one of the pins in -to set is not an endpoint in a timing analysis sense. This flag is shown only if a to set is specified in the exception.
- o: overridden exception
  In this case the exception specification is met by some timing path but another exception is applicable to the path (due to a higher exception precedence or more constraining value).

If a path is unconstrained (that is, if it has a required time of infinity), timing exceptions do not change the path constraints. Applying a timing exception to an unconstrained path does not change the path constraint. A path is unconstrained if it has an infinite required time.

To remove timing exceptions from specified paths, use the **reset_path** command. The **reset_design** command removes all attributes from the design, including timing exceptions.

The number of timing exception objects reported in the *from*, *through*, or *to* sets are limited by the value of the **collection_result_display_limit** variable. As in collection result display, setting the variable to a negative value would print the complete set. Any nonnegative value would truncate any set when the size is greater than the value; the truncated output is designated by an ellipsis. Note that for negative settings, the behavior matches object collection query echoing the entire object set. It is recommended to avoid negative settings when reporting large numbers of exceptions with sizable object sets as this can adversely affect the performance of the

report_exceptions command.

The max_time_borrowed report is generated for global reporting only and not when any of the **-from**, **-through**, or **-to** options are specified with the command.

EXAMPLES

The following example lists all timing exceptions set on the design.

```
pt_shell> report_exceptions

****************************************
Report : exceptions
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From          To         Setup          Hold                Ignored
-------------------------------------------------------------------------------
RESET         *          FALSE          FALSE
*             CO         max=4.5        min_rise=2,min_fall=2.5
ffa           ffb        cycles=2       -
```

The following example shows rise/fall qualified timing exceptions set on the design.

```
pt_shell> report_exceptions

From          To         Setup          Hold                Ignored
-------------------------------------------------------------------------------
RESET         *          FALSE          FALSE
*             CO(r)      max=4.5        min_rise=2,min_fall=2.5
ffa(r)        ffb(f)     cycles=2       -
```

The following example lists all ignored timing exceptions set on the design.

```
pt_shell> report_exceptions -ignored

****************************************
Report : exceptions
         -ignored
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From          To         Setup          Hold                Ignored
-------------------------------------------------------------------------------
```

```
QA            *          max=10         -                   f
*             A          cycles=2       -                   t
*             B          FALSE          FALSE               t


Object                   Type   Attributes
-------------------------------------------------------------------------------
CLK                      clock  max_time_borrow=2.3
```

The following example lists some user-entered compressed exceptions.

```
pt_shell> report_exceptions

****************************************
Report : exceptions
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From          Through        To         Setup          Hold          Ignored
-------------------------------------------------------------------------------
{ ffa/CP ffb/CP }
              *              { ffa/D ffb/D ffc/D }     -
                                         max=2                        p
```

SEE ALSO

```
current_design(2)
set_false_path(2)
set_max_delay(2)
set_max_time_borrow(2)
set_min_delay(2)
set_multicycle_path(2)
collection_result_display_limit(3)
```

report_extract_supply_parasitics_options

Reports the current settings of the options specified by **set_extract_supply_parasitics_options** for the **extract_supply_parasitics** command.

SYNTAX

report_extract_supply_parasitics_options

ARGUMENTS

None

DESCRIPTION

The command reports the extraction options that are set for running the **extract_supply_parasitics** command. The current extraction options may have been manually set by the **set_extract_supply_parasitics_options** command or have been loaded from the TLUPlus options saved in the Milkyway design by IC Compiler. These options include map file, tlu plus file, temperature, para directory, num of processes, and extraction type.

EXAMPLES

The following example reports the extraction options read from the Milkyway design:

```
pr_shell> open_mw_cel -lib my_lib my_cel
pr_shell> report_extract_supply_parasitics_options

****************************************
Report : extract_supply_net_options
Design : design
Version: I-2013.12-DEV
Date   : Thu May 23 16:34:27 2013
****************************************

map file: ./map_new
tlu plus file: ./tlu.plus
temperature:    0.000C
para directory: ./RAIL_DATABASE/pgext/para
num proceses: 2
extraction type: RC
```

```
1
```

SEE ALSO

```
set_extract_supply_parasitics_options(2)
extract_supply_parasitics(2)
```

report_extraction_hierarchy

Reports extraction hierarchy for the specified supply nets.

SYNTAX

```
report_extraction_hierarchy
   supply_nets
```

Data Types

supply net collection *supply_nets*

ARGUMENTS

supply_nets

Lists the supply nets to be reported. Each element of the list can be a supply net collection or a name pattern.

DESCRIPTION

This command reports extraction hierarchy for the spcified supply nets. The design must be loaded into the session using the **open_mw_cel** command before **report_extraction_hierarchy** is executed.

The extraction hierarchy is reported for each of the specified supply nets. For hard macros, type of macro model such as CSF or DWM-lite is also reported.

EXAMPLES

The following example shows how to report extraction hierarchy for the supply nets VDD and VSS.

```
pr_shell> report_extraction_hierarchy [list VDD VSS]

****************************************
Report : Extraction Hierarchy
Design : test_chip
Version: G-2012.06
Date   : Mon Jun 18 13:44:05 2012
****************************************
```

```
Instance   Master     Net      Type (Models) Library
-------------------------------------------------------
design_top design_top VDD CEL /depot/mainlib/Kista_csf/top
  inst4 sram1056x12 VDD CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst2 sram1056x12 VDD CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst3 sram1056x12 VDD CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst1 sram1056x12 VDD CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig

Instance   Master     Net      Type (Models) Library
-------------------------------------------------------
design_top design_top VSS CEL /depot/mainlib/Kista_csf/top
  inst4 sram1056x12 VSS CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst2 sram1056x12 VSS CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst3 sram1056x12 VSS CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig
  inst1 sram1056x12 VSS CONN (CSF) /depot/reflib/railLibrary/DWM/design_top/memlib2_csf_orig

1
```

SEE ALSO

open_mw_cel(2)
extract_supply_parasitics(2)

report_global_slack

Displays slack of specified pins or ports.

SYNTAX

```
string report_global_slack
   [-significant_digits digits]
   [-nosplit]
   [-min]
   [-max]
   [-rise]
   [-fall]
   port_pin_list
```

Data Types

```
digits          integer
port_pin_list   list
```

ARGUMENTS

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. **-nosplit** prevents line splitting and facilitates writing software to extract information from the report output.

-min

Displays the minimum slack.

-max

Displays the maximum slack.

-rise

Displays rise slack.

-fall

Displays fall slack.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins and ports in the current design. Pins of hierarchical cells are not reported.

DESCRIPTION

The **report_global_slack** command displays the slack of specified pins and ports. By default, all pins except those of hierarchical cells and ports of the design are reported. The slack shown in the report for a given pin corresponds to the worst slack among all paths traversing that pin. If a pin is both a timing startpoint and endpoint then its slack will be the worst among all paths either starting and ending at the pin.

To report slacks throughout the current design arrival totals and slacks must be available on all pins, not just at endpoints. To achieve this, set the **timing_save_pin_arrival_and_slack** variable to **true** before issuing the **report_global_slack** command. If the **timing_save_pin_arrival_and_slack** variable has not been set to **true**. the command is set it to **true** and updates the design timing before the command executes.

The default of the variable is **false**. If you intend to use this command, it is recommended that you set the **timing_save_pin_arrival_and_slack** variable to **true** before the first timing update, thus preventing the cost of an additional timing update.

The **-max** and **-min** options are mutually exclusive. The **-rise** and **-fall** options are also mutually exclusive. You can choose any combination from the **-max** or **-min** option and **-rise** or **-fall** option to report a particular slack value.

- If you specify only the **-max** option, but not the **-rise** or **-fall** option, the tool reports both the maximum rise and maximum fall slack values.
- If you specify only the **-min** option, but not the **-rise** or **-fall** option, the tool reports both the minimum rise and minimum fall slack values.
- If you specify only the **-rise** option, but not the **-max** or **-min** option, the tool reports both the maximum rise and minimum rise slack values.
- If you specify only the **-fall** option, but not the **-max** or **-min** option, the tool reports both the maximum fall and minimum fall slack values.
- If you specify neither the **-rise** and **-fall** options nor the **-max** and **-min** options, the tool reports the maximum rise, maximum fall, minimum rise, and minimum fall slack values.

If slack attribute is not available at a pin or port, the tool prints *****.

Note that this command expects that the design has already been timed with the **timing_save_pin_arrival_and_slack** variable set to **true**. If the **timing_save_pin_arrival_and_slack** variable is set to its default of **false**, the **report_global_slack** command can potentially take longer during runtime.

EXAMPLES

The following example generates a report of all slack violations in the current design.

```
pt_shell> report_global_slack -significant_digits 4 -nosplit
```

```
****************************************
Report : report_global_slack
 ...
****************************************

Max_Rise      Max_Fall      Min_Rise      Min_Fall   Point
-------------------------------------------------------------
-1.4869       -1.6330        1.5669        1.7130   ff2/D
-1.4869       -1.6330        1.5669        1.7130   in2/Z
-1.6330       -1.4869        1.7130        1.5669   in2/A
-1.6330       -1.4869        1.7130        1.5669   in1/Z
```

SEE ALSO

```
report_timing(2)
report_default_significant_digits(3)
timing_save_pin_arrival_and_slack(3)
pin_attributes(3)
```

report_global_timing

Reports a top-level summary of the design timing.

SYNTAX

```
status report_global_timing
   [-delay_type max | min]
   [-group group_list]
   [-include include_list]
   [-format format_spec]
   [-output file_name]
   [-separate_non_standard_groups]
   [-separate_all_groups]
   [-significant_digits digits]
   [-pba_mode none | path | exhaustive]
```

Data Types

group_list	list
include_list	list
format_spec	string
file_name	string
digits	integer

ARGUMENTS

-delay_type max | min

Specifies the type of timing delays for which violations are to be reported. By default, both setup (max) or hold (min) violations are reported.

-group *group_list*

Reports only the specified path groups.

-include *include_list*

Specifies a list of additional reporting options. *include_list* is a list that includes one or more of the following reporting options.

- **inter_clock** - Shows the breakdown of violations for combinations of launching and capturing clocks.
- **non_violated** - Shows clock combinations that have no violations. By default, only the worst violating timing path to each endpoint is considered in the report.
- **per_clock_violations** - Takes into account other paths to the same endpoint differentiated by launching, capturing clocks and path groups.

- **scenario_details** - In multi-scenario analysis, shows the breakdown of violations shown in the merged report by scenario.

-format *format_spec*

Specifies the reporting format. *format_spec* follows the format {[narrow | wide] [csv]} The following values are allowed:

- **narrow** (default) - Generates a narrow report.
- **wide** - Generates a wide report.
- **csv** - Generates a CSV report. When using this option, specify the output file name with the **-output** option.

-output *file_name*

Specifies the output file name for CSV report.

-separate_non_standard_groups

Divides the report into subreports separating data for standard path groups and for the following nonstandard path groups: **default**, **async_default**, **clock_gating_default**.

-separate_all_groups

Divides the report into subreports showing data for all path groups which have violations separately.

-significant_digits digits

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13. If you do not use this option, the number of digits is specified by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-pba_mode none | path | exhaustive

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.

DESCRIPTION

The **report_global_timing** command generates a top-level summary of the timing for the design. The report shows the worst negative slack, the sum of all negative slacks, and a number of violating endpoints. By default, each violating endpoint is only counted one time and only one worst slack at each violating endpoint contributes to the total negative slack. With the **-group** option, each endpoint contributes to the worst negative slack and total negative slack of each group in which it has a violating slack.

The summary information is broken down into several columns, showing violations for register-to-register, input-to-register, input-to-output, and register-to-output timing paths.

In the distributed analysis mode, only the **-full_design** option is supported at the master process. If you do not

specify any option at the master, the **-full_design** options is enabled by default. All other options are supported with the command when executed on remote workers as part of the distributed command **remote_execute -partitions**.

EXAMPLES

The following example generates a report of violations in the current design for path groups whose name starts with 'clk'.

```
pt_shell> report_global_timing -group [get_path_group CLK*]
****************************************
Report : global_timing
        -group CLK1 CLK0
 ...
****************************************

Setup violations
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6887.98  -6882.84      0.00     -3.17     -1.96
NUM        398       392         0         4         2
-----------------------------------------------------------

Hold violations
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS      -3.55     -3.55      0.00      0.00      0.00
TNS    -561.17   -561.17      0.00      0.00      0.00
NUM        418       418         0         0         0
-----------------------------------------------------------
```

The second example shows a (truncated) report subdivided into subreports for all launching clock and capturing clock combinations. Each subreport shows a C2C line identifying the clock combination. The launching clock name is shown first, followed by '->' and the name of the capturing clock. The symbol "*" means any clock.

```
pt_shell> report_global_timing -include {inter_clock}
****************************************
Report : global_timing
        -include {inter_clock}
 ...
****************************************

Setup violations
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
C2C * -> *
WNS     -26.57    -26.57      0.00     -1.72     -0.98
TNS  -33377.09 -33369.93      0.00     -5.19     -1.96
NUM       1958      1950         0         6         2
-----------------------------------------------------------
C2C * -> CLK0
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6889.55  -6884.41      0.00     -3.17     -1.96
NUM        403       397         0         4         2
-----------------------------------------------------------
```

```
C2C CLK0 -> CLK0
WNS     -20.34    -20.34      0.00      0.00      0.00
TNS   -6884.25  -6884.25      0.00      0.00      0.00
NUM        396       396         0         0         0
------------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS      -1.10      0.00      0.00     -1.10     -0.98
TNS      -3.06      0.00      0.00     -1.10     -1.96
NUM          3         0         0         1         2
------------------------------------------------------------
 ...
```

In the third example, **-include {per_clock_violations}** is added. Comparing the report output from the second example, more violations are reported because more violating paths to the same endpoint are taking into account.

```
pt_shell> report_global_timing -include {inter_cl per_cl}
****************************************
Report : global_timing
        -include { inter_clock per_clock_violations }
 ...
****************************************

Setup violations
-------------------------------------------------------------
          Total   reg->reg    in->reg   reg->out    in->out
-------------------------------------------------------------
C2C * -> *
WNS      -26.57     -26.57      -2.17      -1.72      -0.98
TNS  -391171.34 -391157.25      -2.17      -9.81      -2.05
NUM       57766      57751          1         11          3
-------------------------------------------------------------
C2C * -> CLK0
WNS      -20.34     -20.34       0.00      -1.72      -0.98
TNS   -39827.79  -39817.95       0.00      -7.79      -2.05
NUM        5256       5244          0          9          3
-------------------------------------------------------------
C2C CLK0 -> CLK0
WNS      -20.34     -20.34       0.00       0.00       0.00
TNS    -6884.25   -6884.25       0.00       0.00       0.00
NUM         396        396          0          0          0
-------------------------------------------------------------
C2C ADC_PCMCLK_IN -> CLK0
WNS       -0.09       0.00       0.00       0.00      -0.09
TNS       -0.09       0.00       0.00       0.00      -0.09
NUM           1          0          0          0          1
-------------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS       -1.10       0.00       0.00      -1.10      -0.98
TNS       -3.06       0.00       0.00      -1.10      -1.96
NUM           3          0          0          1          2
-------------------------------------------------------------
C2C ADC_SCLK_IN -> CLK0
WNS       -1.19       0.00       0.00      -1.19       0.00
TNS       -1.19       0.00       0.00      -1.19       0.00
NUM           1          0          0          1          0
-------------------------------------------------------------
C2C C_D950/C_PCU/C_DPCUB_DP/C_RIDP/I15/I_0_0_15/Q -> CLK0
WNS       -7.26      -7.26       0.00       0.00       0.00
TNS    -1775.25   -1775.25       0.00       0.00       0.00
NUM         350        350          0          0          0
-------------------------------------------------------------
 ...
```

The following example shows the use of alternative reporting format styles. The report is simultaneously printed out in a wide format and written into a CSV file:

```
pt_shell> report_global_timing  -format {csv wide} -output ./example.csv
****************************************
Report : global_timing
        -format { wide csv }
        -output example.csv
 ...
****************************************

Setup violations
          Total        |         reg->reg      |     in->reg   |      reg->out   |      in->out    |
 WNS       TNS    NUM |  WNS       TNS   NUM  | WNS  TNS  NUM | WNS    TNS  NUM | WNS    TNS  NUM |
-----------------------------------------------------------------------------------------------------
-26.57 -33377.09 1958 | -26.57 -33369.93 1950 | 0.00 0.00   0 | -1.72 -5.19  6  | -0.98 -1.96   2 |

Hold violations
          Total        |         reg->reg      |     in->reg   |      reg->out   |      in->out    |
 WNS       TNS    NUM |  WNS       TNS   NUM  | WNS  TNS  NUM | WNS   TNS  NUM  | WNS    TNS  NUM |
-----------------------------------------------------------------------------------------------------
-10.89   -757.51  532 | -10.89   -757.51  532 | 0.00 0.00   0 | 0.00  0.00  0   |  0.00  0.00   0 |

pt_shell> sh cat ./example.csv
Type,Total_WNS,Total_TNS,Total_NUM,reg2reg_WNS,reg2reg_TNS,reg2reg_NUM,in2reg_WNS,in2reg_TNS,
in2reg_NUM,reg2out_WNS,reg2out_TNS,reg2out_NUM,in2out_WNS,in2out_TNS,in2out_NUM
max,-26.57,-33377.09,1958,-26.57,-33369.93,1950,0.00,0.00,0,-1.72,-5.19,6,-0.98,-1.96,2
min,-10.89,-757.51,532,-10.89,-757.51,532,0.00,0.00,0,0.00,0.00,0,0.00,0.00,0
```

The following example shows the report with separate results for each path group:

```
pt_shell> report_global_timing -group {CLK0 AUDIO_SCLK_IN} \
            -separate_all_groups -delay_type max
 ...

Setup violations for paths in CLK0
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6887.98  -6882.84      0.00     -3.17     -1.96
NUM        398       392         0         4         2
-----------------------------------------------------------

Setup violations for paths in AUDIO_SCLK_IN
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS      -2.31     -2.31      0.00      0.00      0.00
TNS     -47.60    -47.60      0.00      0.00      0.00
NUM         32        32         0         0         0
-----------------------------------------------------------
```

The following example shows the report with separate results for standard and nonstandard path groups:

```
pt_shell> report_global_timing -group {CLK0 AUDIO_SCLK_IN *async*} \
            -separate_non_standard_groups  -delay_type max
 ...

Setup violations for standard paths
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
```

```
TNS   -6935.58  -6930.45      0.00     -3.17     -1.96
NUM        430       424         0         4         2
-----------------------------------------------------------

Setup violations for paths in **async_default**
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -23.71    -23.71      0.00      0.00      0.00
TNS   -1010.10  -1010.10      0.00      0.00      0.00
NUM        186       186         0         0         0
-----------------------------------------------------------
```

If you specify the **-pba_mode exhaustive** option, path-based analysis is used to calculate the violating slack values. This option requires that a path search be performed to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases.

SEE ALSO

report_analysis_coverage(2)
report_timing(2)

report_hard_macro_power_management_mtcmos

2012"

Report the hard macro power management cells' current or library cell information.

SYNTAX

```
status report_hard_macro_power_management_mtcmos
   -cell object_list
   -lib_cell object_list
```

Data Types

```
list object_list
```

ARGUMENTS

-cell *object_list*

Specifies a list of library cell objects. The *object_list* can be a collection of cell objects, cell name list, or cell name patterns(shell style wildcard supported). This option will be available after **calculate_rail_voltage** finished, since the current value of each on-state resistor is available after **calculate_rail_voltage** command.

-lib_cell *object_list*

Specifies a list of library cell objects, library cell names, or library cell name patterns (shell style wildcard supported).

DESCRIPTION

This command generates hard macro power management cell report. When *-cell* specified, this command reports resistance and current value of each on-state resistors in the specified cell. The information of *-cell* option is available after **calculate_rail_voltage**, since the current value of each on-state resistor is available after **calculate_rail_voltage** command. . When *-lib_cell* specified, this command will list the on-state resistors of the specified library cells.

To enable hard macro power management cell support and use this command, the tcl variable **rail_enable_hm_pm_feature** value has to be true.

EXAMPLES

The following examples report all hard macro power management cell masters, the command support shell style wildcard:

```
pr_shell> report_hard_macro_power_management_mtcmos -lib_cell *

pr_shell> report_hard_macro_power_management_mtcmos \
                  -lib_cell [get_lib_cell */*]
```

The report example:

```
****************************************
Report : Hard Macro Switch Cells
Design : top
****************************************

Lib Cell Name: HM_SWITCH
Number of PM devices: 3

   NET1_NAME LAYER1   X1     Y1  NET2_NAME LAYER2   X2     Y2  R-ON
-------------------------------------------------------------------
[ 0]     VSS   31  42630      0       VSSM   31  42630    660 1.000
[ 1]     VSS   31  42630   1280       VSSM   31  42630   1900 2.000
[ 2]     VSS   31  42630   1280       VSSM   31  42630    660 3.000


The following example will report the date of cell SW1.
pr_shell> report_hard_macro_power_management_mtcmos \
                  -cell [get_cell SW1]

pr_shell> report_hard_macro_power_management_mtcmos \
                  -cell SW1

****************************************
Report : Hard Macro Switch Cells
Design : top
****************************************

Cell Name: SW1
Lib Cell Name: HM_SWITCH
   LAYER1   X1      Y1 LAYER2   X2     Y2  ON-RES CURRENT(A)
-------------------------------------------------------------
[ 0]   31  42630      0   31  42630    660 13.200 3.18571e-05
[ 1]   31  42630   1280   31  42630   1900 13.200 3.15757e-05
[ 2]   31  42630   1280   31  42630    660 13.200 3.17173e-05
```

Where the unit of coordinate is Milkyway database unit, the unit of on-state resistance is ohm.

SEE ALSO

load_power_management_mtcmos_locations(2)

report_hierarchy

Reports the reference hierarchy of the current instance or current design.

SYNTAX

```
string report_hierarchy
   [-full]
   [-noleaf]
   [-nosplit]
```

ARGUMENTS

-full

Displays the full hierarchy. By default, components of submodules in multiple locations in an hierarchy are listed only once. An ellipsis (...) indicates the contents of a previously displayed module.

-noleaf

Does not display the leaf library cells.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

Displays the indented reference hierarchy of the current instance or the current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the current design.

EXAMPLES

The following command reports the full hierarchy of the design.

```
pt_shell> report_hierarchy -full

****************************************
Report : hierarchy
Design : middle
****************************************
```

```
        FD2                             tech_lib
        ND2                             tech_lib
        inter
             low
                 NR4                    tech_lib
        low
             NR4                        tech_lib
```

SEE ALSO

```
report_reference(2)
report_cell(2)
```

report_host_usage

Creates a detailed report that describes all host options that you have created.

SYNTAX

```
status report_host_usage
   [-verbose]
   [-nosplit]
   [host_option_names]
```

Data Types

```
host_option_names     list
```

ARGUMENTS

-verbose

Generates a more detailed report.

-nosplit

Does not split lines if columns overflow.

host_option_names

Reports the specified list of host options, which were previously defined by the **set_hosts_options** command. If you do not specify a list of host options, the command reports all host options by default.

DESCRIPTION

The **report_host_usage** command reports information about the specified sets of host options. The command also reports peak memory and CPU usage for the local process and all online distributed processes.

Running the **report_host_usage** command before starting any hosts reports the following basic information:

- Options Name - The name of the host options set.
- Host Name - The host on which the processes are to be launched. If the processes are to be launched on a managed compute resource, the string "farm" is shown.
- Num Processes - The number of processes to be launched.
- Protocol - The type of the communication protocol specified by the **-protocol** option of the

set_host_options command. The default value of the protocol is **auto**.

The **-verbose** option reports the following additional information:

- Action and Command - The actions and associated commands that you can perform for the host options. SUBMIT indicates the command to submit a process for execution. TERMINATE indicates the command to terminate a job that has not started.
- Protocol - The type of the communication protocol that is actually used to communicate between the master process and remote hosts.
- # - The internal instance number of the process. When each slave instance is launched by the **start_hosts** command it is assigned an instance number. This is to allow you to relate the process to the output of the **start_hosts** command.
- Host Name - The name of the actual host on which the process is running.
- Job ID - If the options specified access a managed resource, such as LSF or SGE, this is the ID of the job submitted to the farm. If this value is -1, the job ID either cannot be determined or the process was shutdown in which case the ID is no longer valid. For non-managed resources, a symbol "-" is shown.
- Process ID - The ID of the slave PrimeTime process. If this value is -1, the process ID either could not be determined or the process was shutdown in which case the ID is no longer valid.
- Status - The status of the slave process. The possible values are

```
LAUNCHED      : The process is launched but not ready for use yet.
ONLINE        : The process is online and ready for use.
SHUTDOWN      : The process was shut down.
FATALed       : The process abnormally terminated.
MISSALIGNED   : The master a slave processes are different versions
                of PrimeTime and the slave process is not usable.
SHUTTING_DOWN : The process is shutting down.
```

Usage Limits (Cores)

This section of the report shows details about the cores usage limits applied to each set of host options. The first entry in this table describes the limits on the current local process. The cores limits columns are

- Effective
  This is the effective cores usage limit to be honored by PrimeTime. This value is computed given the user and hardware limits.

If you use the **-verbose** option, the following columns are also shown:

- User-set - The **max_cores** that you set in the corresponding **set_host_options** command.
- Hardware - The number of logical cores on the target hardware.

Memory Usage

This section of the report shows details about the peak memory usage of all online processes. The first entry in this table describes the current local process. The memory usage columns are

- Memory (MB) - The peak memory (in megabytes) usage of the local and all distributed pt_shell processes currently online.

Performance

This section of the report shows details the CPU and Elapsed time measurements (in seconds) of all online processes. The first entry in this table describes the current local process. The performance columns are

- CPU Time (s) - This shows the total CPU time usage for each online process.
- Elapsed Time (s) - This shows the total elapsed time for each online process.

EXAMPLES

In the following example, PrimeTime is launched onto a 16-core machine. Initially, reporting usage shows that the user-specified local process is effectively limited to 4 cores even though no user limit is set. The subsequent **set_host_options** command increases the number of cores to 8.

```
pt_shell> report_host_usage -verbose
****************************************
Report : host_usage
        -verbose
 ...
****************************************


  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ---------------------------------------------------------------------
  (local process)  -                 4         16            4
  ---------------------------------------------------------------------
  Total                                                      4

  Memory usage

  Options Name     #                           Memory (MB)
  ---------------------------------------------------------------------
  (local process)  -                           287.65

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ---------------------------------------------------------------------
  (local process)  -    2                      6

1

pt_shell> set_host_options -max_cores 8
Information: The max_cores limit for the local (current) process has been
  modified by 4 to 8. (CMCR-103)
1
pt_shell> report_host_usage -verbose
****************************************
Report : host_usage
        -verbose
...
****************************************


  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ---------------------------------------------------------------------
  (local process)  -                 8         16            8
  ---------------------------------------------------------------------
  Total                                                      8
```

```
    Memory usage

    Options Name     #                          Memory (MB)
    ---------------------------------------------------------------------
    (local process)  -                          364.57

    Performance

    Options Name     #    CPU Time (s)          Elapsed Time (s)
    ---------------------------------------------------------------------
    (local process)  -    2                     85

  1
```

For more examples, see the man page for the **set_host_options** command.

SEE ALSO

```
remove_host_options(2)
set_host_options(2)
start_hosts(2)
stop_hosts(2)
```

report_ideal_network

Displays information about ports, pins, nets, and cells on ideal networks in the current design.

SYNTAX

```
status report_ideal_network
   [-net]
   [-cell]
   [-load_pin]
   [-timing]
   [object_list]
```

Data Types

```
object_list      list
```

ARGUMENTS

-net

Displays all nets in the specified ideal networks. By default, nets are displayed for all ideal networks.

-cell

Displays all cells in the specified ideal networks. By default, cells are displayed for all ideal networks.

-load_pin

Specifies to display load pins (boundary pins) of the specified ideal networks along with any ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, load pins are displayed for all ideal networks.

-timing

Specifies to display all internal pins (non-source and non-boundary pins) in the specified ideal networks that have ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, internal pins with timing are displayed for all ideal networks.

object_list

Defines a list of source ports or source pins of the specified ideal networks to be displayed. By default, the source pins and source ports for all ideal networks in the current design are displayed.

DESCRIPTION

Displays information about the ideal networks in the current design. If no arguments are specified, all ideal network sources in the current design are displayed. If you specify a list of source pins or ports, the command displays information about the ideal networks emanating from these objects.

The "Latency" and "Transition" columns show the minimum and maximum values set for both rising and falling edges. Set these values with the **set_ideal_latency** and **set_ideal_transition** commands.

EXAMPLES

The following example sets up an ideal network, applies ideal latency and ideal transition values and shows the output of the ideal network report.

```
pt_shell> set_ideal_network {p_in in1}
pt_shell> set_ideal_latency    -min  1.12 in1
pt_shell> set_ideal_transition -max  9.87 in1
pt_shell> set_ideal_latency    -min  1.34 n0_i/Z
pt_shell> set_ideal_transition -rise 5.62 n3_i/B
pt_shell> report_ideal_network -timing -load_pin -cell -net

****************************************
Report : ideal_network
        -timing
        -load_pin
        -net
        -cell
 ...
****************************************

Source ports                    Latency                        Transition
and pins                  Rise            Fall            Rise            Fall
                       min    max      min    max      min    max      min    max
--------------------------------------------------------------------------------
middle/in1             1.12    --      1.12    --       --    9.87      --    9.87
middle/p_in             --     --       --     --       --     --       --     --

Internal pins                   Latency                        Transition
with ideal timing         Rise            Fall            Rise            Fall
                       min    max      min    max      min    max      min    max
--------------------------------------------------------------------------------
n3_i/B                  --     --       --     --      5.62   5.62      --     --
n0_i/Z                 1.34    --      1.34    --       --     --       --     --

Boundary pins                   Latency                        Transition
                          Rise            Fall            Rise            Fall
                       min    max      min    max      min    max      min    max
--------------------------------------------------------------------------------
o_reg3/D                --     --       --     --       --     --       --     --
middle/p_out            --     --       --     --       --     --       --     --

Nets
--------------------------------------------------------------------------------
in1
p_in
p_out
n2
n1
n0

Cells
```

```
--------------------------------------------------------------------------------
n3_i
n2_i
n1_i
n0_i
```

The following example shows the output of the ideal network report on specified objects.

```
pt_shell> report_ideal_network -timing -load_pin -cell -net {in1}

****************************************
Report : ideal_network
        -timing
        -load_pin
        -net
        -cell
 ...
****************************************

Source ports                    Latency                        Transition
and pins                  Rise            Fall            Rise            Fall
                       min    max      min    max      min    max      min    max
--------------------------------------------------------------------------------
middle/in1             1.12    --      1.12    --       --    9.87      --    9.87

Boundary pins                   Latency                        Transition
                          Rise            Fall            Rise            Fall
                       min    max      min    max      min    max      min    max
--------------------------------------------------------------------------------
o_reg3/D                --     --       --     --       --     --       --     --

Nets
--------------------------------------------------------------------------------
in1
```

SEE ALSO

```
remove_ideal_network(2)
report_constraint(2)
set_ideal_latency(2)
set_ideal_network(2)
set_ideal_transition(2)
```

report_integrity_results

Reports all the integrity results available within the session.

SYNTAX

```
string report_integrity_results
   [result_name]
   [-summary]
   [-verbose]
```

Data Types

```
string result_name
```

ARGUMENTS

result_name

Limits the report to include only the integrity result specified by *result_name*. If no *result_name* is specified, the command reports information of all the integrity results in the session.

-summary

Report the sumarized information for each integrity result, including the number of errors in the integrity result.

-verbose

Expands the report to include additional information about each integrity result, including the number of errors in the integrity result, a snapshot of taps when the result is created.

DESCRIPTION

Whenever the run of the integrity checking command **check_supply_net_integrity** is successfully completed, a named integrity result is created for saving the data produced by that command. The **report_integrity_results** command reports the integrity results included in the current session. All integrity results are reported by default if no *result_name* is specified. The report can be limited to a single result by providing an optional *result_name.*

The default report for an integrity result provides the name of the result. The *-verbose* option includes the number of integrity checking errors that are contained in the result.

EXAMPLES

The following example shows a simple report for the MY_NAMED_INTEGRITY_RESULT and INTEGRITY_RESULT_1 results:

```
pr_shell> report_integrity_results
****************************************
Report integrity check results
Design : micro
Version: E-2010.12
Date   : Mon Jan 17 16:07:00 2011
****************************************
   MY_NAMED_INTEGRITYRESULT
   INTEGRITY_RESULT_1
1
```

The following example shows a summarized report that includes information about how many errors are contained within the integrity results:

```
pr_shell> report_integrity_results -summary
****************************************
Report : integrity check results
Design : micro
Version: E-2010.12
Date   : Mon Jan 17 16:07:00 2011
****************************************
   MY_NAMED_INTEGRITY_RESULT
   28 missing vias errors, 2 floating geometry errors

   INTEGRITY _RESULT_1
   2 floating geometry errors
1
```

The following example shows a verbose report that includes information about not only how many errors are contained within the integrity results but also a snapshot of taps when the result is created:

```
pr_shell> report_integrity_results -verbose
****************************************
Report : integrity check results
Design : micro
Version: E-2010.12
Date   : Mon Jan 17 16:07:00 2011
****************************************
   MY_NAMED_INTEGRITY_RESULT
   28 missing vias errors, 2 floating geometry errors
     Taps in the result: int1 int2

   INTEGRITY _RESULT_1
   2 floating geometry errors
     Taps in the result: int3 int4 int5
1
```

SEE ALSO

```
create_taps(2)
current_integrity_result(2)
get_taps(2)
```

```
remove_integrity_result(2)
rename_integrity_result(2)
report_taps(2)
pr_work_dir(3)
tap_contexts(2)
```

report_latch_loop_groups

Reports on latch data pins involved in loops of transparent latches.

SYNTAX

```
string report_latch_loop_groups
   [-of_objects pin_list]
   [-loop_breakers_only]
   [-path_breakers_only]
   [-nosplit]
```

Data Types

```
pin_list     list
```

ARGUMENTS

-of_objects *pin_list*

Specifies a collection of data pins of transparent latches. Only pins of loop groups containing the specified pins are reported. By default, the command reports on all latch data pins involved with transparent latch loops, as well as other latch data pins outside loops with the **is_latch_loop_breaker** pin attribute set to **true**.

-loop_breakers_only

Specifies that the report should contain only pins that are loop breaker latch data pins and should be pins truly contained in transparent latch loops. By default, all transparent latch data pins within the loop group are reported, along with other latch data pins with the **is_latch_loop_breaker** pin attribute set to **true**.

-path_breakers_only

Specifies that the report should contain only pins with the **is_latch_loop_breaker** pin attribute set to **true**. By default, all transparent latch data pins within the loop group are returned in the collections.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all the latch data pins of

intersecting loops is formed. The **report_latch_loop_groups** command analyzes the design and reports on latch data pins involved with loops. The report also includes some latch data pins outside loops. Combinational pins are not included in the results.

Latch data pins outside loops can be reported if the latch is being treated by the tool as a latch loop breaker data pin, even if it is not inside a latch loop. This can happen when either the user requests the pin to be treated this way using the **set_latch_loop_breaker** command, or the tool has selected the pin to be treated as a loop breaker data pin to save runtime. For more information about how this can happen, see the man page of the **timing_through_path_max_segments** variable.

The report lists the name of the latch data pin, a number identifying which latch loop group the pin is part of, and attributes describing if the pin is a loop breaker latch, and why.

The number identifying which latch loop group the pin is part of is an arbitrary number, except that within a report, the latch data pins that are part of the same loop group have the same number. For latch data pins that are not part of a loop group, "NA" is shown instead of a loop group number.

Before using the **report_latch_loop_groups** command, you must set the **timing_enable_through_paths** variable to **true**.

EXAMPLES

The following example uses the **report_latch_loop_groups** command to report latch loops in the design:

```
pt_shell> report_latch_loop_groups
****************************************
Report : latch loop groups
 ...
****************************************
Attributes
   b loop breaker d pin
   p long path breaker d pin, but not in a loop
   u user requested to be a loop breaker using set_latch_loop_breaker
   a user requested to avoid with set_latch_loop_breaker

  Latch                                Latch Loop    Attributes
  D pin                                Group
-------------------------------------------------------------------
DUT/Latch1/D                           NA            pu
DUT/Latch2/D                           1             b
DUT/Latch3/D                           1
DUT/Latch4/D                           2             bu
DUT/Latch5/D                           2             a
```

SEE ALSO

```
report_latch_loop_groups(2)
timing_enable_through_paths(3)
timing_through_path_max_segments(3)
```

report_lib

Reports library information.

SYNTAX

```
string report_lib
   [-nosplit]
   [-timing_arcs]
   [-power_arcs]
   library
   [lib_cell_list]
```

Data Types

```
library              list
lib_cell_list             list
```

ARGUMENTS

-nosplit

Prevents splitting lines if a column overflows. Some of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-timing_arcs

Displays timing arc information. Indicates to list all cell timing arcs.

-power_arcs

Displays power arc information. Indicates to list all cell power arcs.

library

A pattern matching a single library or a collection containing one library. The library must have been read by using the **read_db** command.

lib_cell_list

Displays a list of library cells about which information is reported. The default is to report information about all cells in the technology library. Each element in this list is either a collection of library cells or a pattern that matches library cells in the *library* option.

DESCRIPTION

A library report displays library information including a list of operating conditions, wire load models, and available library cells. The *-timing_arcs* option lists detailed timing arc information for each library cell, while the *-power_arcs* option lists detailed power arc information. Note: Currently, the *-power_arcs* and *-timing_arcs* options are mutually exclusive. The **report_lib** command also indicates if the library is a maximum library in a max/min relationship to a minimum library created by the **set_min_library** command.

To generate a report, the specified library must be loaded into pt_shell.

EXAMPLES

This example shows the default library report.

```
pt_shell> read_db tech_lib.db

pt_shell> report_lib tech_lib

****************************************
Report : library
Library: tech_lib
****************************************

Time Unit                   : 1 ns
Capacitance Unit            : 1 pF
Default Wire Load Mode      : top
Default Wire Load Model     : Not specified.
Default Wire Selection Group: Not specified.

Operating Conditions:

   Name                Process     Temp        Voltage     Tree Type
   ---------------------------------------------------------------------------
   BCCOM                  0.60        0.00        5.25     best_case
   TYPICAL                1.00       25.00        5.00     balanced_case
   WCCOM                  1.50       70.00        4.75     worst_case


Wire Load Models:

   Name
   ---------------------------------------------------------------------------
   small_wl
   medium_wl
   large_wl

Library Cells:

   Attributes:
    b - black box (function unknown)

   Lib Cell  Attributes
   ------------------------------
   AN2
   BUFA
   BUFB
   FF
   IV
```

```
   LTCH
   MUX21
   OR2
```

In this example, timing arc information is shown for a list of library cells.

```
pt_shell> report_lib -timing_arcs tech_lib {AN2 OR2}

****************************************
Report : library
        -timing_arcs
Library: tech_lib
****************************************

Library Cells:

   Attributes:
    b - black box (function unknown)

                            Arc                    Arc Pins
   Lib Cell  Attributes    #  Sense/Type      From        To         When
   ---------------------------------------------------------------------------
   AN2                     0  unate           A           Z
                           1  unate           B           Z

   OR2                     0  unate           A           Z
                           1  unate           B           Z
```

In the example below, power arc information is shown for a list of library cells.

```
pt_shell> report_lib -power_arcs tech_lib {AN2 OR2}

****************************************
Report : library
        -power_arcs
Library: tech_lib
****************************************

Time Unit                                  : 1 ns
Capacitance Unit                           : 1 pF
Pulling Resistance Unit                    : 10ohm
Voltage Unit                               : 1 V
Current Unit                               : 1e-06 A
Leakage Power Unit                         : 1e-06 W

Scaling Factors:

    *Global*
    ---------------------------------------------------------------------------
    k_process_rise_propagation             :          0.000000
    k_volt_rise_propagation                :          0.000000
    k_temp_rise_propagation                :          0.000000
    k_process_fall_propagation             :          0.000000
    k_volt_fall_propagation                :          0.000000
    k_temp_fall_propagation                :          0.000000
    k_process_rise_transition              :          0.000000
    k_volt_rise_transition                 :          0.000000
    k_temp_rise_transition                 :          0.000000
    k_process_fall_transition              :          0.000000
    k_volt_fall_transition                 :          0.000000
    k_temp_fall_transition                 :          0.000000
    k_process_cell_rise                    :          0.000000
```

```
k_volt_cell_rise                        :           0.000000
k_temp_cell_rise                        :           0.000000
k_process_cell_fall                     :           0.000000
k_volt_cell_fall                        :           0.000000
k_temp_cell_fall                        :           0.000000
k_process_internal_power                :           0.000000
k_volt_internal_power                   :           0.000000
k_temp_internal_power                   :           0.000000
k_process_cell_leakage_power            :           0.000000
k_volt_cell_leakage_power               :           0.000000
k_temp_cell_leakage_power               :           0.000000

AN2_factors
---------------------------------------------------------------------------
k_process_rise_propagation              :           0.000000
k_volt_rise_propagation                 :           0.000000
k_temp_rise_propagation                 :           0.000000
k_process_fall_propagation              :           0.000000
k_volt_fall_propagation                 :           0.000000
k_temp_fall_propagation                 :           0.000000
k_process_rise_transition               :           0.000000
k_volt_rise_transition                  :           0.000000
k_temp_rise_transition                  :           0.000000
k_process_fall_transition               :           0.000000
k_volt_fall_transition                  :           0.000000
k_temp_fall_transition                  :           0.000000
k_process_cell_rise                     :           1.000000
k_volt_cell_rise                        :          -0.440000
k_temp_cell_rise                        :           0.002120
k_process_cell_fall                     :           1.000000
k_volt_cell_fall                        :          -0.440000
k_temp_cell_fall                        :           0.002120
k_process_internal_power                :           0.000000
k_volt_internal_power                   :           0.000000
k_temp_internal_power                   :           0.000000
k_process_cell_leakage_power            :           0.000000
k_volt_cell_leakage_power               :           0.000000
k_temp_cell_leakage_power               :           0.000000

OR2_factors
---------------------------------------------------------------------------
k_process_rise_propagation              :           0.000000
k_volt_rise_propagation                 :           0.000000
k_temp_rise_propagation                 :           0.000000
k_process_fall_propagation              :           0.000000
k_volt_fall_propagation                 :           0.000000
k_temp_fall_propagation                 :           0.000000
k_process_rise_transition               :           0.000000
k_volt_rise_transition                  :           0.000000
k_temp_rise_transition                  :           0.000000
k_process_fall_transition               :           0.000000
k_volt_fall_transition                  :           0.000000
k_temp_fall_transition                  :           0.000000
k_process_cell_rise                     :           1.000000
k_volt_cell_rise                        :          -0.440000
k_temp_cell_rise                        :           0.002120
k_process_cell_fall                     :           1.000000
k_volt_cell_fall                        :          -0.440000
k_temp_cell_fall                        :           0.002120
k_process_internal_power                :           0.000000
k_volt_internal_power                   :           0.000000
k_temp_internal_power                   :           0.000000
k_process_cell_leakage_power            :           0.000000
k_volt_cell_leakage_power               :           0.000000
```

```
    k_temp_cell_leakage_power                :           0.000000

Operating Conditions:

    Name                Process     Temp        Voltage     Tree Type
    ---------------------------------------------------------------------------
    <lib_default>         1.00       27.00        2.50     balanced_case
    B                     0.63      -30.00        2.80     balanced_case
    BB                    0.63      -30.00        2.80     best_case
    BG                    0.63      -30.00        2.75     balanced_case
    TYPICAL               1.00       27.00        2.50     balanced_case
    W                     1.37      125.00        2.20     balanced_case
    WGSM                  1.37       85.00        2.25     balanced_case
    WW                    1.37      125.00        2.20     worst_case

Input Voltages:

    No input_voltage groups specified.

Output Voltages:

    No output_voltage groups specified.


Wire Loading Model:

    No wire loading specified.


Wire Loading Model Mode: top.

Delay Threshold Trip-Points
    input_threshold_pct_rise                 : NOT SPECIFIED
    output_threshold_pct_rise                : NOT SPECIFIED
    input_threshold_pct_fall                 : NOT SPECIFIED
    output_threshold_pct_fall                : NOT SPECIFIED


Slew Threshold Trip-Points
    slew_lower_threshold_pct_rise            : NOT SPECIFIED
    slew_upper_threshold_pct_rise            : NOT SPECIFIED
    slew_lower_threshold_pct_fall            : NOT SPECIFIED
    slew_upper_threshold_pct_fall            : NOT SPECIFIED

    slew_derate_from_lib                     : NOT SPECIFIED


Power Supply Group:

    No power supply group specified.

default_cell_leakage_power                   : 0
default_leakage_power_density                : NOT SPECIFIED

Power Information:

    Attributes:
       a - average power specification
       i - internal power
       l - leakage power
      rf - rise and fall power specification

                                      Power          Source
    Cell         Attributes  #        Toggling pin   of path  When
```

```
--------------------------------------------------------------------------------
AN2          l           0
             i,rf        1        A                        !B
             i,rf        2        B                        !A
             i,rf        3        Z              A         B
             i,rf        4        Z              B         A
             i,a         5        Z
OR2          l           0
             i,rf        1        B                        A
             i,rf        2        A                        B
             i,rf        3        Z              B         !A
             i,rf        4        Z              A         !B
             i,a         5        Z
```

SEE ALSO

read_db(2)
set_min_library(2)

report_lib_groups

Generates a report of library groups.

SYNTAX

```
int report_lib_groups
   -scaling
   [-show]
   [-nosplit]
   [show_list]
```

Data Types

```
show_list       list
```

ARGUMENTS

-scaling

Reports the list of library groups defined for scaling. Scaling library groups can be defined using the **define_scaling_lib_group** command.

-show

Reports the list of options in the *show_list* option. Specify this option if you want detailed information for the libraries in the scaling group, such as voltage and temperature. If this option is used, you must also specify the *show_list* option.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

show_list

Reports the specific options that need to be reported. The *show_list* option can only contain options from the list {temperature, voltage, extended_name}. You can use any combination of these options. The *show_list* option cannot be empty. If the *-show* option is specified, you must also specify a non-empty *show_list*.

DESCRIPTION

Generates a report of the scaling library groups specified in the design.

The *scaling* option allows you to view all the scaling library groups specified using the **define_scaling_lib_group** command.

You can obtain detailed information about the libraries by using the *-show* option. The *-scaling* option is required.

EXAMPLES

The following example shows the scaling relationships that exist in the design.

```
pt_shell> report_lib_groups -scaling
****************************************
Report : lib_groups
         -scaling
Design : test
Version: Y-2006.06-Alpha-DEV
Date   : Wed Mar  8 11:22:51 2006
****************************************

Group    Library
----------------------------------------------------------------
Group 1
         lib_0.95
         lib_0.85

1
```

The following example shows how you can use the *-show* option to obtain more detailed library information.

```
pt_shell> report_lib_groups -scaling -show {voltage temperature}
****************************************
Report : lib_groups
         -scaling
         -show
Design : test
Version: Y-2006.06
Date   : Wed Mar  8 11:50:28 2006
****************************************

Group    Library      Temperature   Voltage
----------------------------------------------------------------
Group 1
         lib_0.95     125.00        0.95
         lib_0.85     125.00        0.85

1
```

SEE ALSO

```
define_scaling_lib_group(2)
report_lib(2)
```

report_min_period

Displays minimum-period check information about specified pins or ports.

SYNTAX

```
string report_min_period
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-path_type format]
   [-input_pins]
   [-derate]
   [-nets]
   [-crosstalk_delta]
   [-transition_time]
   [-capacitance]
   [port_pin_list]
```

Data Types

digits	integer
format	string
port_pin_list	list

ARGUMENTS

-all_violators

Reports only violating minimum period checks.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-nosplit

Prevents line splitting in the report.

-path_type *format*

Specifies the format of the path report and how the clock path is displayed. Allowed values are

- **summary** (the default) - Displays one line for each path that shows only the required period, actual period, and slack.
- **short** - Shows only startpoints and endpoints in the clock path.

- **full_clock** - Shows full clock paths.
- **full_clock_expanded** - Shows full clock paths including all master clocks of a generated clock.

-input_pins

Reports the input pins. The default is to show only output pins.

-derate

Reports the derating factors. The default is to show no derate factors. This also shows the derate factor specified on the required period if one has been specified by the **set_timing_derate** command. Note that this option applies only when the **-path_type** option is set to either the **full_clock** or **full_clock_expanded** options.

-nets

Reports nets. The default is not to show nets.

-crosstalk_delta

Reports the annotated delta delay and delta transition time. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the **set_annotated_delay -delta_only** and **set_annotated_transition -delta_only** commands. Note that the **-crosstalk_delta** option only reports the calculated or annotated deltas; it does not initiate crosstalk analysis. Only deltas on input pins are shown. Delta transition time is shown only with the **-transition_time** option. The **-crosstalk_delta** option automatically sets the **-input_pins** option.

-transition_time

Reports the transition time (slew). The default is not to show transition time. For each driver pin or load pin the transition time is displayed in a column preceding incremental path delay.

-capacitance

Reports the total (lump) capacitance. The default is not to show capacitance. When the **-nets** option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

port_pin_list

Reports the specified list of pins or ports. If you do not use this option, the report contains all pins and ports in the current design.

DESCRIPTION

The **report_min_period** command displays the following information for the minimum period check: required period, actual period, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator.

If the **timing_remove_clock_reconvergence_pessimism** variable is set to **true**, clock reconvergence pessimism is removed from the slack on the reports.

To show more detailed minimum-period calculations, use the **-path_type** and **-input_pins** options.

When used under simultaneous multivoltage analysis (SMVA) analysis mode it will report only the cross-domain paths for minimum period checks.

You can also report minimum-period violations by using the **report_constraint** command with the **-min_period** option.

EXAMPLES

The following example generates a report of all minimum-period constraints in the current design.

```
pt_shell> report_min_period

****************************************
Report : min period
        -path_type summary
 ...
****************************************

   sequential_clock_min_period

                            Required        Actual
   Pin                     min period     min period        Slack
   ---------------------------------------------------------------
   b1/dst/CK (CK1 rise)         2.90           1.64          -1.26  (VIOLATED)
   b1/dst/CK (CK1 fall)         2.90           1.80          -1.10  (VIOLATED)
   b2/dst/CK (CK2 rise)         2.90           3.60           0.70  (MET)
   b2/dst/CK (CK2 fall)         2.90           3.80           0.90  (MET)
```

The following example generates a report of the minimum-period violation including the full clock path trace and the derate factor for the block1/test/CK pin.

```
pt_shell> report_min_period -path_type full_clock -derate block1/test/CK
****************************************
Report : min period
        -path_type full_clock
        -derate
 ...
****************************************

  Pin: block1/test/CK
  Related clock: CK2
  Check: sequential_clock_min_period

  Point                                 Derate    Incr       Path
  ---------------------------------------------------------------
  clock CK2 (fall edge)                            9.00       9.00
  clock source latency                             0.00       9.00
  CK2 (in)                                         0.00       9.00 f
  u2/Z (IBUF1)                           1.00      0.87 H     9.87 f
  block2/u1/Z (IBUF1)                    1.00      0.97 *    10.83 f
  block2/u2/Z (IBUF1)                    1.00      1.78 *    12.61 f
  block2/test/CK (NT9P32K1PG)            1.00      1.50 *    14.11 f
  open edge clock latency                                    14.11

  clock CK2 (fall edge)                           12.00      12.00
  clock source latency                             0.00      12.00
  CK2 (in)                                         0.00      12.00 f
  u2/Z (IBUF1)                           1.00      0.77 H    12.77 f
  block2/u1/Z (IBUF1)                    1.00      0.87 *    13.63 f
  block2/u2/Z (IBUF1)                    1.00      1.68 *    15.31 f
  block2/test/CK (NT9P32K1PG)            1.00      1.50 *    16.81 f
  clock uncertainty                               -0.20      16.61
```

```
close edge clock latency                                    16.61
-----------------------------------------------------------------
required min period                    1.00                  2.90 *
actual period                                                2.50
-----------------------------------------------------------------
slack (VIOLATED)                                            -0.40
```

SEE ALSO

```
report_constraint(2)
report_min_pulse_width(2)
report_timing(2)
report_default_significant_digits(3)
timing_remove_clock_reconvergence_pessimism(3)
```

report_min_pulse_width

Displays minimum pulse width check information about specified pins or ports.

SYNTAX

```
string report_min_pulse_width
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-path_type format]
   [-input_pins]
   [-derate]
   [-nets]
   [-crosstalk_delta]
   [-transition_time]
   [-capacitance]
   [port_pin_list]
```

Data Types

digits	integer
format	string
port_pin_list	list

ARGUMENTS

-all_violators

Reports only violating minimum pulse width checks.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-nosplit

Prevents line splitting in the report.

-path_type *format*

Specifies the format of the path report and how the clock path is displayed. Allowed values are:

- **summary** (the default) - Shows one line for each path and only the required pulse width, actual pulse width, and slack
- **short** - Shows only startpoints and endpoints in the clock path

- **full_clock** - Shows full clock paths
- **full_clock_expanded** - Shows full clock paths including all master clocks of a generated clock

-input_pins

Indicates that input pins are shown in the path report. The default is to show only output pins.

-derate

Indicates that derate factors are shown in the path report. The default is to show no derate factors. This also shows the derate factor specified on the required pulse width if one has been specified by the **set_timing_derate** command. Note that this option applies only when the **-path_type** option is set to either the **full_clock** or **full_clock_expanded** options.

-nets

Indicates that nets are to be shown in the path report. The default is not to show nets.

-crosstalk_delta

Reports the annotated delta delay and delta transition time. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the **set_annotated_delay -delta_only** and **set_annotated_transition -delta_only** commands. Note that the **-crosstalk_delta** option only reports the calculated or annotated deltas; it does not initiate crosstalk analysis. Only deltas on input pins are shown. Delta transition time is shown only with the **-transition_time** option. The **-crosstalk_delta** option automatically sets the **-input_pins** option.

-transition_time

Shows the transition time (slew). The default is not to show transition time. For each driver pin or load pin the transition time is displayed in a column preceding incremental path delay.

-capacitance

Shows the total (lump) capacitance. The default is not to show capacitance. When the **-nets** option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

port_pin_list

Reports only the specified list of pins or ports. If you do not use this option, the report shows all pins and ports in the current design.

DESCRIPTION

The **report_min_pulse_width** command displays the following information for the minimum pulse width check: required pulse width, actual pulse width, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator.

If the **timing_remove_clock_reconvergence_pessimism** variable is set to true, clock reconvergence pessimism is removed from the slack on the reports.

More detailed minimum pulse width calculations are shown using the **-path_type** and **-input_pins** options.

Minimum pulse width violations can also be reported with the **report_constraint** command using the **-min_pulse_width** option.

Note that if the **timing_enable_pulse_clock_constraints** variable is set to **true** (default value), the **report_min_pulse_width** command does not report pulse width checks in the pulse clock networks. To report pulse width checks in the pulse clock network use the **report_pulse_clock_min_width** command. To check pulse width in the pulse clock network using the **report_min_pulse_width** command, set the **timing_enable_pulse_clock_constraints** variable to **false**.

When used under simultaneous multivoltage analysis (SMVA) analysis mode it will report only the cross-domain paths for minimum pulse width checks.

EXAMPLES

The following example generates a report of all minimum pulse width violations in the current design.

```
pt_shell> report_min_pulse_width

****************************************
Report : min pulse width
        -path_type summary
 ...
****************************************

   sequential_clock_pulse_width

                   Required       Actual
   Pin             pulse width    pulse width      Slack
   ---------------------------------------------------------------
   ff1/CP            10.20          10.00          -0.20  (VIOLATED)
   ff2/CP            10.20          10.04          -0.16  (VIOLATED)
   ff3/CP             2.10           2.00          -0.10  (VIOLATED)
   ff3/CP             2.10           2.00          -0.10  (VIOLATED)
   ff2/CP            10.00           9.96          -0.04  (VIOLATED)


   clock_tree_pulse_width

                   Required       Actual
   Pin             pulse width    pulse width      Slack
   ---------------------------------------------------------------
   nand1/Z           10.00           9.58          -0.42  (VIOLATED)
   or1/B             10.00           9.58          -0.42  (VIOLATED)
   nand1/A           10.20          10.00          -0.20  (VIOLATED)
   or1/Z             10.20          10.04          -0.16  (VIOLATED)
   or1/Z             10.00           9.96          -0.04  (VIOLATED)
```

SEE ALSO

```
remove_min_pulse_width(2)
report_constraint(2)
report_pulse_clock_min_width(2)
report_timing(2)
set_min_pulse_width(2)
set_pulse_clock_min_width(2)
timing_enable_pulse_clock_constraints(2)
report_default_significant_digits(3)
timing_remove_clock_reconvergence_pessimism(3)
```

report_mode

Displays a report of modes for specified cells or design.

SYNTAX

```
string report_mode
   [-type cell | design]
   [-nosplit]
   [instance_list]
```

Data Types

```
instance_list          list
```

ARGUMENTS

-type design | cell

Indicates the type of mode to be reported. This option has the mutually exclusive valid values of either **design** or **cell**.

The *cell* value specifies that cell modes are to be reported. Cell modes are defined on library cells in the library.

The *design* value specifies that design modes are to be reported. Design modes are user-defined using the **define_design_mode_group** and **map_design_mode** commands. If the *-type* option is ommitted from the command, this is equivalent to specifying *-type cell*.

-nosplit

Most of the mode information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

instance_list

Specifies a list of instances whose modes are to be reported. By default, the report includes all cells that have modes. This option is only valid if -type has a cell value

DESCRIPTION

This command reports cell modes or design modes. When reporting cell modes, the report specifies whether the cell mode is enabled or disabled, and it also specifies the reason why the mode is enabled or disabled. There are four possible reasons as to why a mode can be enabled or disabled. They are

- cell - Indicates that the cell mode has been set directly using the **set_mode -type cell** command.
- design - Indicates that the cell mode has been set due to the activation of a design mode using the **set_mode -type design** command. The design mode must have previously been mapped to the cell mode using the **map_design_mode** command. The report also specifies the name of the activated design mode. To view more details about the design mode, use the **report_mode -type_design** command.
- cond - Indicates that the cell mode has been set due to evaluation of the mode condition during constant propagation.
- default - Indicates that the cell mode has not been set by any of the above reasons.

When reporting design modes the report displays the design modes specified on the current design and for each design mode, reports the cell modes and moded pins associated with that design mode.

EXAMPLES

The following example reports cell mode information for the design top_design. Two cells have modes, Uram1 and Uram2 are instances of the library cell RAM2_core. rw is a cell mode group defined on the library cell RAM2_core. This cell mode group has two cell modes read and write. No modes are currently active so all modes are enabled and the reason is given as default.

```
pt_shell> report_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                              Mode(Group)       Status      Reason
-------------------------------------------------------------------------------
Uram1/core(RAM2_core)             read(rw)          ENABLED     default
                                  write(rw)         ENABLED     default
-------------------------------------------------------------------------------
Uram2/core(RAM2_core)             read(rw)          ENABLED     default
                                  write(rw)         ENABLED     default
-------------------------------------------------------------------------------
```

The following example shows a report of modes specified for the two RAMs that have modes in the design. Ram Uram1 is set in mode read, and Ram Uram2 is set in mode write. Thus, all timing arcs of RAM Uram1 associated with mode read are enabled , and all timing arcs associated with mode write are disabled.

```
pt_shell> set_mode read Uram1/core
pt_shell> set_mode write Uram2/core
pt_shell> report_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                              Mode(Group)    Status         Reason
--------------------------------------------------------------------------------
Uram1/core(RAM2_core)             read(rw)       ENABLED        cell
                                  write(rw)      disabled       cell
--------------------------------------------------------------------------------
Uram2/core(RAM2_core)             read(rw)       disabled       cell
                                  write(rw)      ENABLED        cell
```

```
---------------------------------------------------------------------------
```

The following example reports that two design modes have been specified using the **set_mode -type design** command. For each design mode, it reports any mapping of cell modes, as specified by the **map_design_mode** command. The report also displays any paths that are design mode dependent, as specified by the **map_design_mode -from** or **map_design_mode -to** command.

```
pt_shell> map_design_mode  {read,latching} {U2/core,U1/core} read
pt_shell> map_design_mode  -from {INFF1/CLK} -to {INFF3/D}
pt_shell> map_design_mode  -from {INFF1/CLK} -to {INFF4/D}
pt_shell> map_design_mode  {write,transparent} {U2/core,U1/core} write
pt_shell> report_mode -type design


****************************************
Report : design_mode
Design : misc
Version: v4.0a-development
Date   : Fri Mar 29 13:29:05 1996
****************************************

-------------------------------------------------------------
Design Mode Group : 'default'
      Design Mode : 'read' (is current design mode)
      -----------------------------------------------------------
         Cell                          Mode(Group)
         -------------------------------------------------------------
         U2/core                       latching(output_latch)
                                       read(rw)
         -------------------------------------------------------------
         U1/core                       latching(output_latch)
                                       read(rw)
         From Pin
         -------------------------------------------------------------
         INFF1/CLK
         INFF1/CLK

         To Pin
         -------------------------------------------------------------
         INFF3/D
         INFF4/D
-------------------------------------------------------------

-------------------------------------------------------------
Design Mode Group : 'default'
      Design Mode : 'write'
      -----------------------------------------------------------
         Cell                          Mode(Config)
         -------------------------------------------------------------
         U2/core                       transparent(output_latch)
                                       write(rw)
         -------------------------------------------------------------
         U1/core                       transparent(output_latch)
                                       write(rw)
-------------------------------------------------------------
```

SEE ALSO

map_design_mode(2)

```
define_design_mode_group(2)
remove_design_mode(2)
reset_mode(2)
set_mode(2)
```

report_mtcmos_switch_cell_sequence

Reports mtcmos switch cell event sequence

SYNTAX

```
string report_mtcmos_switch_cell_sequence
   supply_nets
```

Data Types

```
list supply_nets
```

ARGUMENTS

supply_nets

The event sequence of mtcmos switch cells, which are connected to the supply nets, is reported. The list of supply nets should be the main supply net to which the mtcmos witch cells are connected with main pg pins. If a virtual supply net is specified in the supply_nets list, we will ignore it and issue a warning message.

DESCRIPTION

The **report_mtcmos_switch_cell_sequence** command displays information about the mtcmos switch cell event sequence. For each supply net, this report always shows two sets of switching events: one is derived from VCD, which should be available after running command **update_currents -dynamic** ; the other is the user defined mtcmos cells switching events. If **update_currents** is not run or there is no VCD file, the report will show there is no vcd based switching events for this net. If user does not define switching events for any mtcmos cells with command **create_mtcmos_switch_cell_sequence**, the report will show there is no user defined switching events for this net.

For each of the main supply net specified in the supply_net list, here is what we report:

- The section for VCD based mtcmos switch cell sequence.
- The section for user defined mtcmos switch cell sequence.

In each of the above section report, it shows the switching events cell by cell. Only the events that enable mtcmos switch cells on this net will be reported. For each of the mtcmos switch cells that have switching events, the report shows:

- Cell name, names of input singal pins, names of output signal pins, pg pin name.
- List of events. For each event, it shows:

- Absolute event time, pin state for each of the cell's signal pins, the pin that is switching, and the transition time for that switching pin. The direction of the transition is indicated by that pin's pin state, "r" for rise, "f" for fall.

The time unit and pin state legend are explained in the header section of the report. For VCD based switching events, the transition time on the swiching pin is obtained from timing analysis. For user defined swiching events, the transition time on the switching pin is the user assigned value.

EXAMPLES

The following example shows how to report mtcmos switch cell event sequence:

```
pr_shell> report_mtcmos_switch_cell_sequence TVDD
****************************************
Report : report mtcmos switch cell sequence
Design : design
Version: H-2013.06-BETA2
Date   : Fri Apr 19 11:17:38 2013
****************************************

Units and legend:
----------------------------------------------
Time_unit : 1 second
x  : unknown pin state
1  : pin state is logic 1
0  : pin state is logic 0
r  : pin state is rising from 0 to 1
f  : pin state is falling from 1 to 0

**************************************************
* Report of pm cell switch activity for net TVDD:
**************************************************
[VCD based pm cell switch events] (cell name: {event_time signal_pin_states})
There is no vcd based pm cell switch events for this net.
[User assigned pm cell switch events] (cell name: {event_time signal_pin_states})
Cell name: u5_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=3.70763e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=3.93727e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
Cell name: u4_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=3.44728e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=4.27864e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
Cell name: u3_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=3.17288e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=4.67569e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
Cell name: u2_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=2.90271e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=5.06549e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
Cell name: u1_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=2.65471e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=5.46034e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
Cell name: u0_HEADER2.  Input signal pins: switch_in2 switch_in1.  Output signal pins: switch_out2 sv
 {time=2.50005e-09  "switch_in1"=0 "switch_in2"=r "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
 {time=5.87662e-09  "switch_in1"=r "switch_in2"=1 "switch_out1"=x "switch_out2"=x  Switching_Pin="swi
```

SEE ALSO

```
create_mtcmos_switch_cell_sequence(2)
remove_mtcmos_switch_cell_sequence(2)
```

report_mtcmos_switch_lib_cell

Report MTCMOS switch library cells and their related library information set by the command **set_mtcmos_switch_lib_cell.**

SYNTAX

```
status report_mtcmos_switch_lib_cell
   cell_master_list
```

Data Types

```
list cell_master_list
```

ARGUMENTS

cell_master_list

Specifies a list of MTCMOS switch cell master names such that their related library information is reported. Items within the list can only be cell master names.

DESCRIPTION

The command provides a report of library information on MTCMOS switch cell masters set through the command **set_mtcmos_switch_lib_cell.** This command can only be issued before the command **open_mw_cel.** The command will fail if it is issued after the command **open_mw_cel.**

The default information reported for a MTCMOS switch library cell includes pin state, pin state name, input PG pin, output PG pin, on state linear resistance and off state leakage. (See the **set_mtcmos_switch_lib_cell** command for how to specify pin state name, on state resistance, and off state leakage.)

EXAMPLES

The following example sets pin state names, an on state resistance and an off state leakage current for cell master HEADER, then uses report_mtcmos_switch_lib_cell to report library information for cell master HEADER:

```
pr_shell> set_mtcmos_switch_lib_cell HEADER \
                -input_pg_pin VDD -output_pg_pin VVDD \
                -pin_state {switch_in 0} -on_state_resistance 1.0 \
                -pin_state_name ON
```

```
pr_shell> set_mtcmos_switch_lib_cell HEADER \
                -input_pg_pin VDD -output_pg_pin VVDD \
                -pin_state {switch_in 1} -off_state_leakage 0.1 \
                -pin_state_name OFF
pr_shell> report_mtcmos_switch_lib_cell HEADER
****************************************
Report : report_mtcmos_switch_lib_cell
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************
Library cell : 'HEADER'

            Pin State  Input   Output  On State    Off state
Pin State   Name       PG Pin  PG Pin  Resistance  Leakage
-------------------------------------------------------------------------------
switch_in 0   ON         VDD     VVDD    1.0         -
switch_in 1   OFF        VDD     VVDD    -           0.1
```

SEE ALSO

set_mtcmos_switch_lib_cell(2)

report_mw_lib

Displays information about the measurement units or reference libraries of a Milkyway library.

SYNTAX

```
status report_mw_lib
   [-unit_range]
   [-mw_reference_library]
   [mw_lib]
```

Data Types

```
mw_lib     string
```

ARGUMENTS

-unit_range

Displays information of about the measurement units for the Milkyway library (length, time, capacitance, and so on) and the characteristics of the routing layers. This option requires you to specify the Milkyway library name.

-mw_reference_library

Reports a list of the reference libraries for the specified Milkyway design library, or for the current Milkyway library if no library name is specified in the command. You define the association between a design library and are reference library with the **set_mw_lib_reference** command.

mw_lib

Specifies the Milkyway library to be reported.

DESCRIPTION

This commands displays information about the measurement units or reference libraries of the current Milkyway library or a specified Milkyway library. Use the appropriate option, either **-unit_range** or **-mw_reference_library**, to specify the type of information to report.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays a list of reference libraries for the current Milkyway design library:

```
prompt> report_mw_lib -mw_reference_library
 ../ref/mw_lib/sc
 ../ref/mw_lib/io
 ../ref/mw_lib/ram8x64
 ../ref/mw_lib/ram16x128
1
```

The following example displays information about the measurement units and routing layers of a specified Milkyway design library:

```
prompt> report_mw_lib -unit_range my_mw_lib_A

Library: my_mw_lib_A
Tech. Attr.   Unit       Resolution   Min. Value    Max. Value
===========   ========   ==========   ==========   ===========
length        micron           1000        0.001   2147483.647
time          ns               1000        0.001   2147483.647
capacitance   pf           10000000    0.0000001   214.7483647
resistance    kohm         10000000    0.0000001   214.7483647

Layer: POLY
Mask name: poly
Attribute                 Minimum    Maximum
======================   ========   ========
thickness                 0.0e+00    0.0e+00
unit resistance           0.0e+00    0.0e+00
unit capacitance          0.0e+00    0.0e+00
 ...

1
```

SEE ALSO

close_mw_lib(2)
open_mw_lib(2)
set_mw_lib_reference(2)

report_name_mapping

Reports the user-defined name-mapping rules.

SYNTAX

```
string report_name_mapping
   [-nosplit]
```

ARGUMENTS

-nosplit

Prevents the line splitting if column overflows.

DESCRIPTION

The **report_name_mapping** command reports the user-defined name-mapping rules you have set using the **set_rtl_to_gate_name** command.

EXAMPLES

In the following example, all the user-defined name-mapping rules are reported.

```
pt_shell> set_rtl_to_gate_name -rtl a_net -gate n272
1
pt_shell> set_rtl_to_gate_name -rtl a_pin -gate U3/D0
1
pt_shell> set_rtl_to_gate_name -rtl a_cell -gate U3
1
pt_shell> report_name_mapping

****************************************
Report : report_name_mapping
Design : mac
Version: D-2010.06-Alpha2
Date   : Fri Apr  2 11:12:53 2010
****************************************

      Attributes
      ----------
```

```
        v - Inverted

    ------------------------------------------------------------------------------
    RTL Name                      Gate Level Name               Type     Attributes
    ------------------------------------------------------------------------------
    a_cell                        U3                            cell
    a_pin                         U3/A                          pin
    a_net                         n39                           net
    ------------------------------------------------------------------------------
```

SEE ALSO

set_rtl_to_gate_name(2)

report_net

Generates a report of net information.

SYNTAX

```
string report_net
   [-connections]
   [-verbose]
   [-significant_digits digits]
   [-segments]
   [-nosplit]
   [net_names]
```

Data Types

```
digits        int
net_names     list
```

ARGUMENTS

-connections

Indicates that the report shows net connection information.

-verbose

Indicates that the report shows verbose connection information. This must be used with the *-connections* option.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which the default value is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h, the upper bound on a meaningful value of the argument to the *-significant_digits* option is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-segments

Indicates that the report shows all global segments for each net requested. Global net segments are all those physically connected across all hierarchical boundaries.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

net_names

Specifies a list of nets that is reported. The default is to report all nets in the current instance or current design.

DESCRIPTION

The command displays information about the nets in the design of the current instance, or in the current design. If the **current_instance** command is set, the report generates for the design of that instance; otherwise, the report generates for the current design. Attributes, such as annotated resistance and annotated capacitance, are displayed.

If the *-connections* optin is used, PrimeTime lists the leaf cell pins connected to each net.

EXAMPLES

The following example reports all nets in the design.

```
pt_shell> report_net

****************************************
Report : net
Design : top
Version: 1997.01-development
Date   : Wed Aug  7 09:28:05 1996
****************************************

Attributes:
    c - annotated capacitance
    r - annotated resistance

Net                  Fanout     Fanin       Cap   Resistance    Pins   Attributes
-------------------------------------------------------------------------------
a                         2         1      2.00         0.00       3
b                         1         1      1.00         0.00       2
c                         1         1      1.00         0.00       2
d                         1         1      1.00         0.00       2
e                         1         1      1.00         0.00       2
en                       20         1     25.00         0.00      21
f                         1         1      1.00         0.00       2
g                         1         1      1.00         0.00       2
h                         1         1      1.00         0.00       2
i1                        1         1      1.00         0.00       2
i2                        1         1      1.00         0.00       2
i3                        1         1      1.00         0.00       2
i4                        1         1      1.00         0.00       2
j                         1         1      1.00         0.00       2
k                         1         1      1.00         0.00       2
l                         1         1      1.00         0.00       2
m                         2         1      2.00         0.00       3
n                         1         1      1.00         0.00       2
o                         1         1      1.00         0.00       2
o1                        1         1      0.00         0.00       2
o2                        1         1      0.00         0.00       2
p                         1         1      1.00         0.00       2
q                         2         1      2.00         0.00       3
```

```
r                        1         1      1.00         0.00       2
s                        1         1      1.00         0.00       2
t                        1         1      1.00         0.00       2
u                        1         1      1.00         0.00       2
w                        1         1      1.00         0.00       2
y                        1         1      1.00         0.00       2
z                        1         1      2.00         0.00       2
-------------------------------------------------------------------------------
Total 30 nets           52        30     56.00         0.00      82
Maximum                 20         1     25.00         0.00      21
Average               1.73      1.00      1.87         0.00    2.73
```

The following example displays a verbose connection report for net z.

```
pt_shell> report_net -verbose -connections z

****************************************
Report : net
        -connections
        -verbose
Design : top
Version: 1997.01-development
Date   : Wed Aug  7 09:30:38 1996
****************************************


Connections for net 'z':
    pin capacitance:    2
    wire capacitance:   0
    total capacitance:  2
    wire resistance:    0
    number of drivers:  1
    number of loads:    1
    number of pins:     2

    Driver Pins              Type                  Pin Cap
    ------------             ----------------      --------
    z/Z                      Output Pin (NOR2)      0

    Load Pins                Type                  Pin Cap
    ------------             ----------------      --------
    o2/B                     Input Pin (BUF)        2
```

SEE ALSO

```
current_design(2)
current_instance(2)
report_cell(2)
report_design(2)
```

report_noise

Reports noise analysis information.

SYNTAX

```
int report_noise
   [-above]
   [-below]
   [-low]
   [-high]
   [-nworst_pins pin_count]
   [-significant_digits digits]
   [-verbose]
   [-nosplit]
   [-slack_lesser_than slack_limit]
   [-slack_type area | height | area_percent]
   [-all_violators]
   [-data_pins]
   [-clock_pins]
   [-async_pins]
   [object_list]
```

Data Types

pin_count	integer
digits	integer
slack_limit	float
object_list	list

ARGUMENTS

-above

Performs the reporting only above the rails. If this option is combined with the *-low* option, it reports for the noise bumps above the low rail. If it is combined with the *-high* option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above the high rail and above the low rail.

-below

Performs the reporting only below the rails. If this option is combined with the *-low* option, it reports for the noise bumps below the low rail. If it is combined with the *-high* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below the high rail and below the low rail.

-low

Performs the reporting only for the low rail. If this option is combined with the *-above* option, it reports the noise bumps above the low rail. If it is combined with the *-below* option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps for both below and above the low rail.

-high

Performs the reporting only for the high rail. If this option is combined with the *-above* option, it reports the noise bumps above the high rail. If it is combined with the *-below* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps for both below and above the high rail.

-nworst_pins *pin_count*

Specifies the number of load pins to be reported. Any number greater than 1 is accepted; the default value is 1.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-verbose

Shows more details about the calculation of total noise on each load pin, including the individual contribution of each aggressor as well as noise bumps propagated from previous stages of the design.

-nosplit

If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

-slack_lesser_than *slack_limit*

Indicates that only those pins with a slack less (more negative) than *slack_limit* are to be shown.

-slack_type area | height | area_percent

Specifies the type of slack to be used. Valid values are *area*, *height*, and *area_percent*. The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin. A *slack_type* value of *height* reports noise slack as the voltage margin. This setting is the default. A *slack_type* value of *area* reports noise slack as the voltage margin multiplied by the noise bump width. A *slack_type* value of *area_percent* reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

-all_violators

Indicates that only violating pins (negative slack) are to be shown. This option cannot be used with the *-slack_lesser_than* option. If this option is used with the *-nworst_pins* option, the number of violating pins will be limited by that value.

-data_pins

Indicates that the reporting is done only on pins that are data pins of sequential cells. The effect is similar to preselect the data pins using all_registers -data_pins and pass the resulting collection to the report_noise command.

-clock_pins

Indicates that the reporting is done only on pins that are clock pins of sequential cells. The effect is similar to preselect the clock pins using all_registers -clock_pins and pass the resulting collection to the report_noise command.

-async_pins

Indicates that the reporting is done only on the asynchronous pins of sequential cells. The effect is similar to preselect the asynchronous pins using all_registers -async_pins and pass the resulting collection to the report_noise command.

object_list

Specifies the load pins for which the noise reporting is performed. If no pin is specified, reporting is performed on the entire design.

DESCRIPTION

This command provides a report of noise information for the current design and reports the top *-nworst_pins* or specified pins given by *object_list* in increasing order of their slack values. By default, the slack values used are based on *-slack_type height*. Using the *-slack_type area* option may result in a different order of pins reported. Pins with slack reported as POSITIVE are not reported in any particular order.

By default, the non-verbose report is generated and shows the total crosstalk noise bump height and width on the load pins. If the *-verbose* option is used, the individual contribution of each effective aggressor and also the amount of propagated noise is also reported.

Beginning with the F-2011.06 release, the *-slack_type* option defaults to *height* instead of *area* since it is a more direct representation of the noise violation on a pin.

EXAMPLES

The following example generates a report for the worst total noise bump above the low rail in the current design.

```
pt_shell> report_noise -above -low
```

The following example generates a report for the five worst total noise bump above the low rail in the current design.

```
pt_shell> report_noise -above -low -nworst_pins 5
```

The following example generates a report for the worst total noise bump above the low rail and also the worst total noise bump below the low rail in the current design.

```
pt_shell> report_noise -low
```

The following example generates a report for the worst total noise bump for each of the four cases of above the low rail, below the low rail, above the high rail, and below the high rail in the current design.

```
pt_shell> report_noise -low -high
```

SEE ALSO

```
update_noise(2)
report_default_significant_digits(3)
```

report_noise_calculation

Displays the actual calculation of noise information for the specified net arc.

SYNTAX

```
string report_noise_calculation
   [-above]
   [-below]
   [-low]
   [-high]
   [-significant_digits digits]
   [-nosplit]
   -from from_pin
   -to to_pin
```

Data Types

```
digits       integer
from_pin     list
to_pin       list
```

ARGUMENTS

-above

Specifies a noise calculation for above ground or power rails noise analysis region.

-below

Specifies a noise calculation for below ground or power rails noise analysis region.

-low

Specifies a noise calculation for ground rail noise.

-high

Specifies a noise calculation for power rail noise.

-significant_digits *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

-from ***from_pin***

Specifies the startpoints of a net arc within a design. The *from_pin* value must be a driver pin or port.

-to ***to_pin***

Specifies the endpoints of a net arc within a design. The *to_pin* value must be the load pin or port of the same net.

DESCRIPTION

Displays details of the noise calculation for the specified net arc. This command uses the noise analysis settings for the design set by the **set_noise_parameters** command. It also performs a noise update if needed.

The report shows some general information about the victim, as well as details of noise bump and noise slack calculations.

The general information section includes the victim driver net, pin, noise composite aggressor mode, and also any possible derating factors set by the **set_noise_derate** command.

The noise calculation section includes the total noise bump as well as the estimated noise bumps induced from each of the aggressors. Note that individual aggressor noise bumps are reported only to show their relative contributions. The linear summation of all individual noise bumps might not match the total noise bump due to the nonlinear behavior of the coupled clusters. The aggressor attribute column shows more details about each of the effective aggressors. Aggressors which are filtered are not included in this report. The following is the list of possible attributes an aggressor can have:

Attributes:

```
A - aggressor is active
C - aggressor is a composite aggressor
D - aggressor is analyzed with detailed engine
E - aggressor is screened due to user exclusion
G - aggressor is analyzed with gate level simulator
I - aggressor has infinite window
L - aggressor is screened due to logical correlation
S - aggressor is screened due to small bump height
X - aggressor is screened due to aggressor exclusion
```

Active aggressor is an aggressor which contributes to the worst case alignment scenario of all the aggressors.

Composite aggressor models the combined effect of a group of weak aggressors to improve runtime and memory capacity.

Aggressor can be screened from the analysis if it has no significant impact on the victim. For example, when the aggressor is excluded by using the **case_analysis** or **set_si_delay_analysis** commands, or if the bump height that it induces on the victim is very small.

Aggressor is analyzed using CCS noise gate level simulation engine if the amplitude of the total noise glitch is significant compared to the lowest noise immunity of the load cells. In this mode, all aggressor drivers as well as the victim driver are analyzed using CCS noise advanced engine and this attribute 'G' shows up on the total noise line.

An aggressor has infinite window when it has no fixed timing relationship with the victim, e.g. an aggressor which is driven by a clock asynchronous to the clock that drives the victim.

Aggressor has logical correlation if there is a common point in the transitive fanin of both aggressor and the victim, or two aggressors.

Aggressor can be excluded from the analysis by using the **case_analysis** or **set_si_delay_analysis** commands.

The noise slack calculation section shows the details of how the area slack and height slack is derived for the victim, and also what kind of constraint was used to derive the slack. The height slack is the difference of the actual bump height versus the bump height which causes the failure with the same width. The area slack is the area that needs to be added to the actual bump to cause a noise failure.

EXAMPLES

The following example specifies a noise calculation report for the driver pin buf2/ZN and load pin buf5/I for below the high rail noise region:

```
pt_shell> report_noise_calculation -below -high -from buf2/ZN -to buf5/I

****************************************
Report : noise_calculation
        -from buf2/ZN
        -to buf5/I
        -high
        -below
 ...
****************************************

Units:  1ns  1pF  1kOhm

Analysis mode                       : report_at_source
Region                              : below_high
Victim driver pin                   : buf2/ZN
Victim driver library cell          : mylib/INVD3
Victim net                          : I2
Victim driver effective capacitance : 0.200827

Steady state resistance source      : library set CCS noise iv curve
Driver voltage swing                : 1.080000
Noise derate height offset          : 0.000000
Noise derate height scale factor    : 1.000000
Noise derate width scale factor     : 1.000000
Noise effort threshold              : 0.000000
Noise composite aggressor mode      : disabled

Noise calculations:

Attributes:
  A - aggressor is active
  C - aggressor is a composite aggressor
  D - aggressor is analyzed with detailed engine
  E - aggressor is screened due to user exclusion
  G - aggressor is analyzed with gate level simulator
  I - aggressor has infinite window
  L - aggressor is screened due to logical correlation
  S - aggressor is screened due to small bump height
  X - aggressor is screened due to aggressor exclusion
```

```
                        Height    Width     Area      Aggressor Attributes
-----------------------------------------------------------------------------
 Aggressors:
 I1                     0.300604  0.786466  0.118207  A I D
 I3                     0.300604  0.786466  0.118207  A I D
 Total:                 0.497124  0.919737  0.228612  G

Noise slack calculation:

Constraint type: library CCS noise immunity

                        Height              Area
-------------------------------------------------------------------
 Required               0.581570            (0.581570 * 0.919737)  -
 Actual                 0.497124            (0.497124 * 0.919737)
-------------------------------------------------------------------
 Slack                  0.084445            0.077668
```

SEE ALSO

```
report_noise(2)
set_noise_derate(2)
set_noise_parameters(2)
set_si_aggressor_exclusion(2)
update_noise(2)
si_noise_composite_aggr_mode(3)
```

report_noise_parameters

Reports status of the noise analysis parameters for the current design.

SYNTAX

```
int report_noise_parameters
```

ARGUMENTS

None.

DESCRIPTION

This command reports status of the parameters that are considered during the noise analysis. If this command is performed, the settings of the parameters for the noise analysis are reported: beyond the rail analysis, arrival times of aggressors and propagation of noise.

EXAMPLES

The following example shows the report format:

```
pt_shell> report_noise_parameters
report_noise_parameters
****************************************
Report  : noise_parameters
Design  : top
Version : Y-2006.06-VA-STA-Beta1-DEV
Date    : Thu Jan 26 09:46:43 2006
****************************************

 ignore arrival       : true
 include beyond Rails : true
 enable propagation   : true
```

SEE ALSO

```
update_noise(2)
report_noise(2)
set_noise_parameters(2)
reset_noise_parameters(2)
si_noise_effort_threshold_within_rails(3)
si_noise_effort_threshold_within_rails(3)
```

report_noise_violation_sources

Reports noise violation sources for failing endpoints.

SYNTAX

```
int report_noise_violation_sources
   [-above]
   [-below]
   [-low]
   [-high]
   [-nworst_endpoints pin_count]
   [-max_sources_per_endpoint pin_count]
   [-significant_digits digits]
   [-verbose]
   [-nosplit]
   [-slack_type slack_type]
   [object_list]
```

Data Types

pin_count	integer
digits	integer
slack_type	float
object_list	list

ARGUMENTS

-above

Performs the reporting only above the rails. If this option is combined with the *-low* option, it reports for the noise bumps above the low rail. If it is combined with the *-high* option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above the high rail and above the low rail.

-below

Performs the reporting only below the rails. If this option is combined with the *-low* option, it reports for the noise bumps below the low rail. If it is combined with the *-high* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below the high rail and below the low rail.

-low

Performs the reporting only for the low rail. If this option is combined with the *-above* option, it reports the noise bumps above the low rail. If it is combined with the *-below* option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps for both below and above the low rail.

-high

Performs the reporting only for the high rail. If this option is combined with the *-above* option, it reports the noise

bumps above the high rail. If it is combined with the *-below* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps for both below and above the high rail.

-nworst_endpoints *pin_count*

Specifies the number of endpoint pins to be reported. Any number greater than 1 is accepted; the default value is 1.

-max_sources_per_endpoint *pin_count*

Specifies the number of violation source pins per endpoint to be reported. Any number greater than 1 is accepted; the default value is 1.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-verbose

Shows details about pins in fan-in cone of the logic from violation sources to endpoint. Pins are ordered from the endpoint to sources and levelized by the distance from the endpoint. The worst slacks are shown on the top.

-nosplit

If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

-slack_type *slack_type*

Specifies the type of slack to be used. Valid values are *area*, *height*, and *area_percent*. The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin. A *slack_type* value of *height* reports noise slack as the voltage margin. This setting is the default. A *slack_type* value of *area* reports noise slack as the voltage margin multiplied by the noise bump width. A *slack_type* value of *area_percent* reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

object_list

Specifies the load pins for which the noise reporting is performed. If no pin is specified, reporting is performed on the entire design.

DESCRIPTION

Provides a report of noise violation source information for failing noise endpoints of the current design.

A noise endpoint is one of the followings: - Data pin, clock pin, or asynchronous pin of flip-flops - Input pin of level sensitive latch - Output port - Load pin of gates where noise exceeds the threshold

A failing endpoint is an endpoint with negative slack value. A violation source is the source of violation that causes a failure in the endpoint.

This command reports the violation sources of the specified or the nworst endpoints.

By default, the non verbose report is generated which shows the total crosstalk noise bump height and width as well as slack on the load pins.

Beginning with the F-2011.06 release, the *-slack_type* option defaults to *height* instead of *area* since it is a more direct representation of the noise violation on a pin.

EXAMPLES

The following example generates a report for the worst violation source of the worst endpoint above the low rail in the current design.

```
pt_shell> report_noise_violation_sources -above -low
```

The following example generates a report for the worst violation source of five worst endpoints above the low rail in the current design.

```
pt_shell> report_noise_violation_sources -above -low -nworst_pins 5
```

The following example generates a report for the worst violation source above the low rail and also the worst violation source below the low rail in the current design.

```
pt_shell> report_noise -low
```

The following example generates a report for the worst violation source for each of the four cases of above the low rail, below the low rail, above the high rail, and below the high rail in the current design.

```
pt_shell> report_noise -low -high
```

SEE ALSO

update_noise(2)
set_noise_parameters(2)
report_noise(2)
report_default_significant_digits(3)

report_ocvm

Reports information about AOCV or POCV derating tables and coefficients; shows details about path-based and graph-based calculations.

SYNTAX

```
status report_ocvm
   [-early]
   [-late]
   [-rise]
   [-fall]
   [-clock]
   [-data]
   [-voltage voltage_value]
   [-cell_delay]
   [-net_delay]
   [-lib_cell]
   [-list_annotated]
   [-list_not_annotated]
   [-coefficient]
   -type aocvm | pocvm
   [-nosplit]
   [object_list]
```

Data Types

voltage_value	string
object_list	list

ARGUMENTS

-early

Shows objects annotated with the AOCV or POCV early deratings.

-late

Shows objects annotated with the AOCV or POCV late deratings.

-rise

Shows objects annotated with the AOCV or POCV rising deratings.

-fall

Shows objects annotated with the AOCV or POCV falling deratings.

-clock

Shows objects annotated with the AOCV or POCV clock deratings.

-data

Shows objects annotated with the AOCV or POCV data deratings.

-voltage *voltage_value*

Shows objects annotated with the AOCV or POCV voltage deratings.

-cell_delay

Shows objects annotated with the AOCV or POCV cell deratings.

-net_delay

Shows objects annotated with the AOCV or POCV net deratings.

-lib_cell

This option can be used with either the -list_annotated or -list_not_annotated options to report the library cell name instead of the instance name.

-list_annotated

Lists the leaf cells and global nets that are annotated with AOCV or POCV derating tables.

-list_not_annotated

Lists the leaf cells and global nets that are not annotated with AOCV or POCV derating tables.

-coefficient

Shows the AOCV or POCV coefficients. You can specify a collection of lib_cell, lib_timing_arc, and cell objects in the *object_list* to display coefficients annotated only on those objects.

-type aocvm | pocvm

Specifies the OCV type. The allowed values are

- **aocvm** - Shows AOCV information.
- **pocvm** - Shows POCV information.

-nosplit

Prevents split lines when columns overflow.

object_list

Specifies either timing_path objects for which path-based metrics are to be reported; or timing_arc objects for which graph-based metrics are to be reported; or design objects on which AOCV or POCV derating tables and coefficients have been annotated.

DESCRIPTION

This command displays the user-specified AOCV or POCV information. The type of information displayed by PrimeTime depends on the type of AOCV or POCV information annotated.

EXAMPLES

SEE ALSO

```
report_aocvm(2)
timing_aocvm_enable_analysis(3)
timing_pocvm_enable_analysis(3)
```

report_path_group

Reports path_group information.

SYNTAX

```
string report_path_group
   [-nosplit]
   [path_group_list]
```

Data Types

```
path_group_list          list
```

ARGUMENTS

-nosplit

Does not split lines if a column overflows.

path_group_list

Displays the path group information for the *path_group_list* that is specified.

DESCRIPTION

Produces a report showing information about path groups in the **current_design** command. The report includes path groups automatically created by the **create_clock** command and those manually created with the **group_path** command. Path groups are used to affect the calculation of the Maximum Delay cost during optimization. The cost of each group is displayed using the **report_constraint** command.

EXAMPLES

The following command displays all the path groups in the current design.

```
pt_shell> report_path_group

****************************************
Report : path_group
Design : test
```

```
Version: A-2007.12-DEV
Date   : Sun Jul 29 12:03:16 2007
****************************************

Path_Group     Weight   From      Through        To
-------------------------------------------------------
**default**      1.00   -         -              -
C1               1.00   *         *              C1
C2               1.00   *         *              C2
CLK              1.00   *         *              CLK
```

SEE ALSO

```
create_clock(2)
current_design(2)
group_path(2)
report_constraint(2)
reset_design(2)
```

report_pg_pin_mapping

Reports PG pin mapping information for cells, cell masters or library cells.

SYNTAX

```
status report_pg_pin_mapping
   object_list
```

Data Types

```
list object_list
```

ARGUMENTS

cell_master_list

Specifies a list of objects whose related PG pin mapping information is to be reported. Items within the list can be cell master names, cells or library cells.

DESCRIPTION

The command reports the PG pin mapping information on a list of objects, either derived from the library or set with the **set_pg_pin_mapping** command.

When this command is issued before the **open_mw_cel** command, the object list can only contain cell master names. The command reports PG pin information that is set with the **set_pg_pin_mapping** command.

When this command is issued after the **open_mw_cel** command, the object list can be either a collection of cells or library cells. The command reports the PG pin mapping information from the current session.

EXAMPLES

The following example sets and reports the PG pin mapping information for the cell masters.

```
pr_shell> set_pg_pin_mapping INV64 \
                -from_ground_pin VSS1 -to_power_pins {VDD1} \
pr_shell> set_pg_pin_mapping LVS32 \
                -from_ground_pin VSS1 -to_power_pins {VDD1 VDD2} \
pr_shell> set_pg_pin_mapping LVS32 \
```

```
                 -from_ground_pin VSS2 -to_power_pins {VDD2} \
pr_shell> report_pg_pin_mapping
****************************************
Report : PG pin mapping
Design : sample_design
Version: D-2013.12
Date   : Fri Feb  5 17:17:46 2013
****************************************
                        Library_cell   Ground_pin          Mapped_power_pins
--------------------------------------------------------------------------------
                        INV64          VSS1                VDD1
                        LVS32          VSS1                VDD1 VDD2
                                       VSS2                VDD2
```

The following example first opens the cell my_cell with the **open_mw_cel** command and then reports PG pin mapping information for all cell instances belonging to the cell master LVS32.

```
pr_shell> open_mw_cel -lib my_design my_cell
pr_shell> report_pg_pin_mapping [get_cell -filter ref_name==LVS32]
****************************************
Report : PG pin mapping
Design : sample_design
Version: D-2013.12
Date   : Fri Feb  5 17:17:46 2013
****************************************
  Cell                  Library_cell   Ground_pin          Mapped_power_pins
--------------------------------------------------------------------------------
  lvs_01                LVS32          VSS1                VDD1 VDD2
                                       VSS2                VDD2
  lvs_02                LVS32          VSS1                VDD1 VDD2
                                       VSS2                VDD2
```

The following example opens the library my_design and the cell my_cell, and then reports the PG pin mapping information a library cell collection.

```
pr_shell> open_mw_cel -lib my_design my_cell
pr_shell> report_pg_pin_mapping [get_lib_cell */LVS32]
****************************************
Report : PG pin mapping
Design : sample_design
Version: D-2013.12
Date   : Fri Feb  5 17:17:46 2013
****************************************
                        Library_cell   Ground_pin          Mapped_power_pins
--------------------------------------------------------------------------------
                        LVS32          VSS1                VDD1 VDD2
                                       VSS2                VDD2
```

SEE ALSO

open_mw_cel(2)
set_pg_pin_mapping(2)

report_pg_pin_only_lib_cell

Reports library information for the PG-pin-only library cells that are created by the **create_pg_pin_only_lib_cell** command.

SYNTAX

status **report_pg_pin_only_lib_cell**
[cell_master_list]

Data Types

list *cell_master_list*

ARGUMENTS

cell_master_list

Specifies the names of PG-pin-only cell masters whose related library information is to be reported. Items within the list can only be cell master names. If no list is given, the command reports all cell masters that are created by the **create_pg_pin_only_lib_cell** command.

DESCRIPTION

The command reports library information on PG-pin-only cell masters that are created with the **create_pg_pin_only_lib_cell** command. Run the command before the **open_mw_cel**/**open_block** command; otherwise the command run will fail.

By default, the command reports the name and types of power, ground and bias pins, and the associated voltages. Each pin name will be followed by pin type. Available pin types are:

Power and Ground pins : :p primary, :b backup, :i internal

Bias pins : :nw nwell, :pw pwell, :dnw deep_nwell, :dpw deep_pwell

EXAMPLES

The following example creates PG-pin-only library cells for the cell master DECAP1, then runs the **report_pg_pin_only_lib_cell** command to check the library information of the PG-pin-only library cells.

```
pr_shell>create_pg_pin_only_lib_cell DECAP1 \
                -power_pins {VDD 1.0} -ground_pins {VSS 0.0} \
pr_shell> create_pg_pin_only_lib_cell DECAP2 \
                -power_pins {VDD 1.0} -ground_pins {VSS 0.0} \
pr_shell>report_pg_pin_only_lib_cell
****************************************

  Lib cell              Power pins          Ground pins          Bias pins
--------------------------------------------------------------------------------
  DECAP1                VDD:p (1.0V)        VSS:p (0V)
  DECAP2                VDD:p (1.0V)        VSS:p (0V)
1
```

SEE ALSO

create_pg_pin_only_lib_cell(2)

report_port

Displays port information within the design.

SYNTAX

```
string report_port
   [-verbose]
   [-design_rule]
   [-drive]
   [-input_delay]
   [-output_delay]
   [-wire_load]
   [-nosplit]
   [port_list]
```

Data Types

```
port_list        list
```

ARGUMENTS

-verbose

Indicates that the port report includes all port information. By default, only a summary section is displayed that lists all ports and their direction.

-design_rule

Reports only port design rule information, including maxCap, manLoad, and maxFanout.

-drive

Reports only drive resistance, input transition time, and driving cell information for only input and inout ports.

-input_delay

Reports only the port input delay information you set.

-output_delay

Reports only the port output delay information you set.

-wire_load

Reports only the port wire load information.

-nosplit

Prevents line splitting if column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

port_list

Displays information on these ports in the current design. Each element in this list is either a collection of ports or a pattern matching the port names.

DESCRIPTION

Displays information about ports in the current design. By default, the command produces a brief report including all ports in the design.

The input and output delay information lists the minimum rise, minimum fall, maximum rise, maximum fall delays, and the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise, the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered to be coming from a level-sensitive latch, or output delay is considered to be going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

Some information, such as whether the port is in an ideal network or not, is displayed after a timing update (as a result of manually executing an **update_timing** function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about ports without incurring the cost of a complete timing update.

EXAMPLES

This example shows the default port report.

```
pt_shell> report_port

****************************************
Report : port
Design : middle
Version: Y-2006.06
Date   : Wed Mar  8 07:26:43 2006
****************************************

Attributes:
    I - ideal network

                        Pin      Wire
Port           Dir      Cap      Cap     Attributes
-------------------------------------------------------
CLOCK          in       0.0000   0.0000
OPER           in       0.0000   0.0000
in0            in       0.0000   0.0000
in1            in       0.0000   0.0000
in2            in       0.0000   0.0000
out_1          out      0.0000   0.0000
out_2          out      0.0000   0.0000
out_3          out      0.0000   0.0000
```

```
out_4           out     0.0000   0.0000
p_in            in      0.0000   0.0000
p_out           out     0.0000   0.0000

1
```

SEE ALSO

```
report_design(2)
report_hierarchy(2)
report_net(2)
report_cell(2)
report_reference(2)
```

report_power

Generates power reports.

SYNTAX

```
string report_power
   [-net_power]
   [-cell_power]
   [-threshold_voltage_group]
   [-lvth_groups low_vth_list]
   [-leaf]
   [-include_boundary_nets]
   [-verbose]
   [-nworst number]
   [-sort_by sort_method]
   [-nosplit]
   [-hierarchy]
   [-levels level]
   [-power_greater_than threshold]
   [-clocks clock_list]
   [-groups group_list]
   [-include_estimated_clock_network]
   [-derate]
   [-rails rails_supply_nets_name]
   [-pattern_priority pattern_list]
   [-attribute attribute_name_for_cell_pattern]
   [-rail_instance_power]
   [-file output_file_name]
   [-pg_pin]
   [object_list]
```

Data Types

low_vth_list	list
number	integer
sort_method	string
level	integer
threshold	float
clock_list	list
group_list	list
rails_supply_nets_name	list
pattern_list	list
attribute_name_for_cell_pattern	string
output_file_name	string
object_list	list

ARGUMENTS

-net_power

Use this option to generate the net-based power report.

-cell_power

Use this option to generate the cell-based power report.

-threshold_voltage_group

Use this option to report the leakage power per voltage threshold group, instead of reporting the number of cells per voltage threshold group using the **report_threshold_voltage_group** command. You can also use this option in conjunction with the standard filtering options such as **-clocks**, **-groups**, and **-rail**.

-lvth_groups *low_vth_list*

Use this option to specify the list of threshold voltage groups to be considered as low threshold voltage. The value of the argument is a list of threshold voltage group identifiers. Each identifier is a non-empty string that might consist of alphanumeric characters and the underscore character ("_"). If this option is omitted, no low threshold voltage groups are identified with the "L" attribute. This option is valid only with **-threshold_voltage_group** option.

-leaf

Use this option to indicate that the power report must traverse the hierarchy and report the nets or cells at lower-levels (as if the design's hierarchy were flat). The default is to report objects at only the current level of hierarchy. When you use this option in combination with the **-levels** option, the lower levels of the design hierarchy are traversed only to the depth you have specified.

-include_boundary_nets

Use this option to indicate that the switching power of primary input nets must be counted when generating the power report; the default is to exclude boundary input nets.

-verbose

Use this option to report verbose information, mainly in the header of the report, such as operating conditions, wire load models used to calculate power and the power unit information.

-nworst *number*

Use this option to filter the report so that it displays only the top-most *number* of the nets or cells sorted by a certain type of sort method.

-sort_by *sort_method*

Use this option to specify the sorting mode for the net or cell order in the power report.

-nosplit

Use this option to prevent line-splitting or section-breaking. By default, if the information for a given field exceeds its fixed column width, the next field begins on a new line, starting in the correct column (line-splitting). By default, if the information for one line exceeds the 80 character limit, the report is broken into two sections, each containing part of the information (section-breaking).

-hierarchy

Use this option to generate the hierarchy-based power report.

-levels *level*

Use this option to specify the number of hierarchy levels to be displayed in the hierarchy-based power report. When you use the **-leaf** option, the number of levels to traverse for the net and cell based report is also specified.

-power_greater_than *threshold*

Use this option to report only the nets or cells with total power value equal to or greater than *threshold* value.

-clocks *clock_list*

Use this option to report only the nets, cells, or both nets and cells that belong to the clock domains specified by the *clock_list* value.

-groups *group_list*

Use this option to report only the nets, cells, or both nets and cells that belong to the power groups specified by the *group_list* value.

-include_estimated_clock_network

Use this option to include the clock network power estimated by the **estimate_clock_network_power** command in the power report.

-derate

Use this option to include the effective power derate factors in the cell reports. It does not affect other types of reports.

-rails *rails_supply_nets_name*

Use this option to specify a list of supply nets under UPF mode or design rails under rail mapping mode for which the power values are reported.

-pattern_priority *pattern_lists*

Specifies the list of patterns of cell name (or its attribute value) to be processed for generating the threshold voltage group leakage report.

-attribute *attribute_name_for_cell_pattern*

Specifies the attribute name, whose values are processed based on the pattern_priority list for generating the threshold voltage group leakage report.

-rail_instance_power

Use this option to report calculated_power , assigned_power , scaled_power and used_power in rail analysis.

-file *output_file_name*

Use this option with 'rail_instance_power' to redirect the report into a file. This will report the domain_clock for each cell as well.

-pg_pin

Use this option with 'rail_instance_power' to report the power details of each pg_pin.

object_list

Use this option to specify a list of cells, nets, or both to be displayed in the cell-based, net-based, or both cell- and net-based power report.

DESCRIPTION

Use the **report_power** command to generate a power report. It can be used in nondistributed power analysis or distributed power analysis. In the distributed power analysis, the command in the master has no options and only generates a merged power report for top design. The following description is for the normal distributed power analysis (nondistributed power analysis or usage in slaves of distributed power analysis).

Generates power reports for the design or a specific hierarchy. The specific hierarchy is determined by the **current_instance** command. Before generating the report, power information is updated if necessary.

Four types of power reports can be generated: summary report, cell-based report, net-based report, and hierarchy-based report.

Summary Power Report

When none of the **-cell_power**, **-net_power**, or **-hierarchy** options are specified, a summary power report is generated. The summary power report displays internal, leakage, switching, and total power, as well as peak power, peak time, glitching power, and X transition power, if available, for the current instance, or the current design if the current instance is the top hierarchy of the current design. The summary power report also reports power for each power group. By default, all instances are placed in only one power group; so the summation of the power of the predefined groups equals the total power consumption of the design.

There are seven predefined power groups:

- io_pad: Cells defined as part of the pad_cell group in the library.
- memory: Cells defined as part of the memory group in the library.
- black_box: Cells with no functional description in the library.
- clock_network: Cells in the clock_network excluding io_pad cells.
- register: Latches and flip-flops driven by the clock network excluding io_pads and black_boxes.
- combinational: Nonsequential cells with a functional description.
- sequential: Latches and flip-flops clocked by signals other than those in the clock network.

See the Power Group Power Reporting section.

To report the total leakage breakdowns, set the **power_report_leakage_breakdowns** variable to **true**.

Cell-Based Power Report

When you specify the **-cell_power** option, the cell-based power is reported. The cell-based power report has two sections: Section one displays the cell internal power, cell leakage power, switching power of nets driven by the cell, total power, and percentage of total power compared with the overall power of all cells displayed in the report.

Section two reports the peak power, peak time, glitching power, and X transition power. Section two only shows up when there are peak power, glitching power, X transition power, or both.

When you specify the **-nosplit** option, the two sections are merged together. The peak power for a cell is only available if the waveform is created for this cell. For more information, see the **update_power** man page.

By default, only cells (including hierarchical cells) in current hierarchy are displayed. The power information displayed for a hierarchical cell is the total effect of all leaf cells contained by the hierarchical cell.

If the **power_report_leakage_breakdowns** variable is set to **true**, an additional section of leakage breakdowns is also be printed out.

If the **-derate** option is specified along with the **-cell_power** option, effective power derate factors are included in the power reports. The effective power derate factors are the ratio of derated power to underrated

power of a cell. Power components with zero power derate factors are excluded from calculating effective power derate factors. In such a case, a 'z' attribute is added to the corresponding cell.

If you use the **set_power_analysis_options** command with the **-sdpd_tracking** option, the report shows an additional attribute flag "s" for those cells whose state-dependent and path-dependent (SDPD) switching activities are tracked.

Net-Based Power Report

When you specify the **-net_power** option, the net-based power is reported. The net-based power report displays VDD, total net load, static probability, toggle rate, and net switching power.

By default, only nets in the current hierarchy are displayed. If you use the **-leaf** option, the report would traverse down all lower hierarchies and report all nets found in each hierarchy. When you specify the *object_list* option, only the nets in the object_list are reported. If a net has more than one net segment in different hierarchies, only the net segment in the top most hierarchy is listed in the report.

Hierarchy-Based Power Report

When you specify the **-hierarchy** option, the hierarchy-based power is reported. The report would be in a hierarchical format, with power information on a block-by-block basis. The hierarchy is shown through indentations. The **-levels** option specifies the number of levels of hierarchy to be displayed. Hierarchy levels deeper than the ones specified in this option are not shown in the report. If the **-leaf** option is specified, the leaf cell reference names are to be reported.

For each hierarchical block, the switching, internal, and leakage power are reported, as well as the total power and the percentage of total power compared with the overall power of the top hierarchy. The top hierarchy is determined by current_instance. Hierarchy-based report also has a second section to display peak power, peak time, glitching power, and X transition power, when necessary. Similarly, if the **-nosplit** option is specified, the two sections are merged together. Peak power for the hierarchy is only available if the waveform is created for this hierarchy.

For more information, see the **update_power** man page.

Rail and Supply Nets Based Power Reports

When you specify the **-rails** option, only the power reports associated with the specified rails or supply nets are generated. The **-rails** option accepts a list of name strings. Each name string can contain one or more rail or power supply net names separated by space. One power report is generated for each name string if it contains valid rails or supply nets. If the name string contains "all" keyword, a report of all rails is generated. Multiple power reports can be generated by using the **report_power** command with different combinations of rail names in the list. The **-rails** option works with other options of the **report_power** command to generate cell, hierarchical, or group-based power reports. For time-based power analysis, peak power reports are always for all rails, regardless of the rail specification.

Negative Power Reporting

When you specify the **-power_greater_than** option along with an appropriate negative threshold value with the **-cell_power** option, the leaf cells with negative power are reported. The default for the threshold is *zero*.

Sorting and Filtering

When you specify a certain sort mode with the **-sort_by** option, the list of cell and nets in the cell- based or net-based power report can be sorted. The available sorting modes for the **-net_power** or **-cell_power** option as following:

```
-net_power option         -cell_power option
-----------------         ------------------
```

```
name                      name
net_static_probability    cell_internal_power
net_switching_power       cell_leakage_power
net_toggle_rate           dynamic_power
total_net_load            total_power
total_power
```

If both the **-net_power** and **-cell_power** options are specified and a sorting mode is explicitly selected, the selected sorting mode is used for both the cell and nets reports. Therefore, you must select a sorting mode that applies to both the **-net_power** and **-cell_power** options.

When reporting with the **-hierarchical** option, the same sorting modes are available as the **-cell_power** option. The report is first sorted by hierarchy and then sorted by the requested sorting method, preserving the hierarchical structure of the report.

If the sorting mode is not explicitly set, a default is chosen based on the mode of the **report_power** command:

```
Mode                          Implicit default
----------                    ------------------
-net_power                    net_switching_power
-cell_power                   cell_internal_power
-net_power -cell_power        dynamic_power
```

The sorted list of the cells and nets in the cell-based and net-based power report can be filtered to display only the top most number of cells or nets specified by the **-nworst** option.

Clock Domain Power Reporting

When you specify the *clock_list* value with the **-clocks** option, a power report is generated for the cells and nets in the specified clock domains.

The cells and nets in the clock tree network of CLK clock are taken as belonging to CLK clock domain. The cells and nets in a timing path launched by the CLK clock are regarded as belonging to CLK clock domain. The cells and nets in a timing path with no launch clock, but captured by the CLK clock are regarded as belonging to the CLK clock domain as well. If a cell or net belongs to multiple clock domains, it is forced to belong to the clock domain with the fastest clock. If a cell or net belongs to no clock domains, it is forced to belong to the fastest clock domain in the design.

The power for the design or a hierarchy is no longer the total effect of all leaf cells in the design or hierarchy, but instead, the total effect of all leaf cells that are in the design or hierarchy AND belong to the specified clock domains.

Clock domain power reporting supports all of the four types of power report.

Power Group Power Reporting

When you specify the *group_list* value with the **-groups** option, a power report is generated for the cells and nets in the specified clock domains.

Power groups can be created by the **create_power_group** command. The tool has also several predefined power groups. For more information, see the **create_power_group** man page.

The power for the design or a hierarchy is no longer the total effect of all leaf cells in the design or hierarchy, but instead, the total effect of all leaf cells that are in the design or hierarchy AND belong to the specified power groups.

Power group power reporting supports all of the four types of power report. In the summary power report, if the **-groups** option is not specified, all the power groups are reported. For each power group, the cumulative switching, internal, and leakage power of the power group are reported, as well as the total power and the

percentage of the total power compared with the overall power of the top hierarchy. The top hierarchy is determined by current_instance. This power group report also has a second section to display the peak power, peak time, glitching power, and X transition power, when necessary.

Similarly, if the **-nosplit** option is specified, the two sections are merged together. Peak power for the power group is only available if the waveform is created for this power group. For more information, see the **update_power** man page.

Clock Tree Power Reporting

Clock tree power and register (driven by the clock tree) power can be reported using the **report_power** command by using the power group power reporting. There are two predefined power groups: clock_network and register. The clock_network power group contains all cells in the clock tree network. The register power group contains all registers driven by the clock tree network.

The **power_clock_network_include_clock_gating_network** and **power_clock_network_include_register_clock_pin_power** variables can affect clock tree power reporting. For more information, see the man pages.

The clock tree power reporting supports all of the four types of power report. By default, the summary power report displays power of the predefined clock_network and register power groups. An attribute label "i" is used to indicate whether the clock_network or register power group includes the register clock pin power. In the cell-based power report, an attribute label "c" is used to indicate that only clock pin internal power is displayed for the register cell; a "d" attribute label is used to indicate that the power displayed for the register cell does not include the register clock pin internal power.

The clock tree power for certain clocks (if not all clocks) can be reported by combining the **-groups** and **-clocks** options.

Estimated Clock Tree Power Reporting

Clock tree power and register (driven by the clock tree) power estimated using the **estimate_clock_network_power** command can be reported using the **-include_estimated_clock_network** option. Estimated clock tree power can only be reported in the Summary Report; therefore, the **-cell_power** and **-net_power** options cannot be specified together with the **-include_estimated_clock_network** option.

The power of the existing clock tree network, if any, is subtracted from the total power, and the power of the estimated clock tree power is added. If the estimated clock tree power includes register power by using the **estimate_clock_network_power -include_registers** command, the power of the existing registers is subtracted from the total power and the power of the estimated register power is added. Accordingly, the power for the power groups "clock_network" and "register" should be affected. These power groups do not include estimated clock network power. However, the power for the existing clock network is subtracted.

A new section in the Summary power report section shows the estimated clock network power for each clock that has been estimated. The **estimate_clock_network_power** command must be used before the **report_power** command. Multiple **estimate_clock_network_power** commands can be used, but only the power from the latest command that estimates clock tree power for a certain clock is reported by the **report_power** command.

The **-clocks** option can be specified together with **-include_estimated_clock_network** option. In this case, only the clock tree power estimated for the specified clocks is included in the power report.

The **-groups** option can be specified together with **-include_estimated_clock_network** option. In this case, only when the *clock_network* and *register* power groups are specified, the estimated clock network power is reported.

Low-Level Hierarchy Power Reporting

Power reports can be generated for the top design and a lower-level hierarchy. The hierarchy can be determined by **current_instance**. The power reports for the top design are special cases, when the current instance is the top-level hierarchy of the design. If the current instance is not the top design, only the cells or nets under the current instance are targeted for power reporting.

To generate different power reports for each clock domain, power group, and hierarchy, combine the **-clocks** and **-groups** options with using the **current_instance** command.

EXAMPLES

The following example shows a verbose summary power report.

```
pt_shell> report_power -verbose
****************************************
Report : power
        -verbose
 ...
****************************************
Library(s) Used: power_lib (File: /remote/libraries/power_lib.db)

Operating Conditions:
Wire Loading Model Mode: enclosed

Cell    Design           Wire Loading Model      Library
------------------------------------------------------------
        mydesign            0.5K_TLM          power_lib.db
sub     submodule           0.5K_TLM          power_lib.db
------------------------------------------------------------

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                 Internal  Switching  Leakage    Total
Power Group      Power     Power      Power      Power     (     %)  Attr
--------------------------------------------------------------------------
io_pad           0.0000    0.0000     0.0000     0.0000    ( 0.00%)
memory           0.0000    0.0000     0.0000     0.0000    ( 0.00%)
black_box        0.0000    0.0000     0.0000     0.0000    ( 0.00%)
clock_network   1.813e-04  4.199e-05  4.192e-10  2.233e-04 (70.03%)  i
register        8.442e-05  1.114e-05  9.208e-09  9.557e-05 (29.97%)
combinational    0.0000    0.0000     0.0000     0.0000    ( 0.00%)
sequential       0.0000    0.0000     0.0000     0.0000    ( 0.00%)
-------------------------------------------------------------------------------

  Net Switching Power  = 5.313e-05   (16.66%)
  Cell Internal Power  = 2.657e-04   (83.33%)
  Cell Leakage Power   = 9.627e-09   ( 0.00%)
                         ---------
Total Power            = 3.188e-04  (100.00%)
```

```
1
```

The following example shows a net-based power report sorted by the *net_switching_power* value and filtered to display only the five nets with highest switching power.

```
pt_shell> report_power -net_power -leaf -nworst 5
****************************************
Report : power
        -net_power
        -nworst 5
        -leaf
        -sort_by net_switching_power
        -power_greater_than       0
 ...
****************************************


  Attributes
  ----------
      a  -  Switching activity information annotated on net
      p  -  Propagated switching activity information on net
      d  -  Default switching activity used on net
      u  -  Net switching activity uninitialized
      m  -  Net is driven by multiple pins

                       Total      Static   Toggle   Switching
Net              Vdd   Net Load   Prob.    Rate     Power      Attrs
----------------------------------------------------------------------
net36            1.80    0.026    0.248    0.1985   8.397e-06  p
net42            1.80    0.026    0.248    0.1985   8.397e-06  p
net48            1.80    0.026    0.248    0.1985   8.397e-06  p
net54            1.80    0.026    0.248    0.1985   8.397e-06  p
sub/net20        1.80    0.026    0.248    0.1985   8.397e-06  p
----------------------------------------------------------------------
Total (5 nets)                                       4.199e-05 Watt

1
```

The following example shows a cell-based power report for clock_network power group (clock tree power).

```
pt_shell> report_power -cell_power -groups clock_network
****************************************
Report : power
        -cell_power
        -sort_by cell_internal_power
        -power_greater_than       0
        -groups clock_network
Design : mydesign
 ...
****************************************
  Attributes
  ----------
      a  -  Annotated internal & leakage power
      b  -  Black box (unresolved) cell
      c  -  Clock pin internal power only
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

              Internal  Switching  Leakage    Total
Cell          Power     Power      Power      Power   (     %)  Attrs
----------------------------------------------------------------------
sub           3.626e-05 8.397e-06 8.384e-11 4.466e-05 (20.00%)  h
clk_out1_reg  1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
clk_temp1_reg 1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
```

```
temp1_reg_3_  5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
temp1_reg_0_  5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
-------------------------------------------------------------------
Totals        1.813e-04 4.199e-05 4.192e-10 2.233e-04 (100.0%)
-------------------------------------------------------------------
1
```

The following example shows a hierarchy-based power report for the "sub" instance.

```
pt_shell> current_instance sub
pt_shell> report_power -hierarchy -levels 2
****************************************
Report : power
        -hierarchy
        -levels 2
Design : mydesign/sub (submodule)
 ...
****************************************

                           Switch   Int      Leak      Total
Hierarchy                  Power    Power    Power     Power    %
-------------------------------------------------------------------
sub (submodule)            9.98e-06 5.33e-05 1.93e-09  6.33e-05 100.0
  clk_gate_out_reg (SNPS_CLOCK_GATE_HIGH_submodule)
                           9.14e-06 1.64e-05 2.41e-10  2.56e-05  40.4
-------------------------------------------------------------------
1
```

The following example shows a report including estimated clock-network power.

```
pt_shell> report_power -include_estimated_clock_network
****************************************
Report : power
 ...
****************************************
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power     (%)    Attrs
--------------------------------------------------------------------------------
io_pad                  0.0000    0.0000     0.0000     0.0000     (0.00%)
memory                  0.0000    0.0000     0.0000     0.0000     (0.00%)
black_box               0.0000    0.0000     0.0000     0.0000     (0.00%)
clock_network           0.0000    0.0000     0.0000     0.0000     (0.00%)
register                8.442e-05 1.114e-05  9.208e-09  9.557e-05 (29.97%)   i
combinational           0.0000    0.0000     0.0000     0.0000     (0.00%)
sequential              0.0000    0.0000     0.0000     0.0000     (0.00%)
-----------------------------------------------------------------------------------
  Attributes
  ----------
      i  -  Including driven register power
                        Internal  Switching  Leakage    Total
Clock                   Power     Power      Power      Power   (%)  Attrs
--------------------------------------------------------------------------------
clk                     1.813e-04 4.199e-05  4.192e-10  2.233e-04
--------------------------------------------------------------------------------
Estimated Clock         1.813e-04 4.199e-05  4.192e-10  2.233e-04(70.03%)
-----------------------------------------------------------------------------------

  Net Switching Power  = 5.313e-05  (16.66%)
  Cell Internal Power  = 2.657e-04  (83.33%)
```

```
  Cell Leakage Power   = 9.627e-09  ( 0.00%)
--------------------------------------------------
Total Power            = 3.188e-04  (100.00%)
--------------------------------------------------
1
```

The following example shows the leakage variation report.

```
pt_shell> report_power
****************************************
Report : Leakage Variation Report
Design : test_design
 ...
****************************************
Library(s) Used: power_variations_lib (File: /remote/libraries/power_variations_lib.db)

Operating Conditions:
Wire Load Model Mode: unknown
(no wire load model is set)

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W

----------------------------------------------------------------------
Design                quantile    sensitivity    mean        stddev
----------------------------------------------------------------------
test_design           1.21771e-06 0.416507    6.25407e-07 2.60486e-07
----------------------------------------------------------------------
1
```

The following example shows power reports for rail VDD1 and VDD2, VDD2 and VDD3 and for all rails.

```
pt_shell> report_power -rails {"VDD1 VDD2" "VDD2 VDD3" "all"}
****************************************
Report : Averaged Power
 ...
****************************************
Current Power Rail: all

Power Report For Rails: VDD1 VDD2

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                    Internal   Switching  Leakage    Total
Power Group         Power      Power      Power      Power      (     %)  Attrs
--------------------------------------------------------------------------------
io_pad              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
memory              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
black_box           0.0000     0.0000     0.0000     0.0000     ( 0.00%)
clock_network       0.0000     0.0000     0.0000     0.0000     ( 0.00%)  i
register            0.0000     0.0000     0.0000     0.0000     ( 0.00%)
combinational       4.061e-06  1.800e-07  8.415e-11  4.241e-06 (100.00%)
sequential          0.0000     0.0000     0.0000     0.0000     ( 0.00%)
--------------------------------------------------------------------------------

  Net Switching Power  = 1.800e-07   ( 4.24%)
  Cell Internal Power  = 4.061e-06   (95.75%)
```

```
  Cell Leakage Power   = 8.415e-11   ( 0.00%)
-------------------------------------------------------
Total Power            = 4.241e-06  (100.00%)
-------------------------------------------------------------
****************************************
Report : Averaged Power
 ...
****************************************

Current Power Rail: all

Power Report For Rails: VDD2 VDD3
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                        Internal   Switching  Leakage    Total
Power Group             Power      Power      Power      Power      (     %)  Attrs
-----------------------------------------------------------------------------------------
io_pad                  0.0000     0.0000     0.0000     0.0000     ( 0.00%)
memory                  0.0000     0.0000     0.0000     0.0000     ( 0.00%)
black_box               0.0000     0.0000     0.0000     0.0000     ( 0.00%)
clock_network           0.0000     0.0000     0.0000     0.0000     ( 0.00%)  i
register                0.0000     0.0000     0.0000     0.0000     ( 0.00%)
combinational           1.973e-06  1.714e-07  4.356e-11  2.145e-06 (100.00%)
sequential              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
-----------------------------------------------------------------------------------------

  Net Switching Power  = 1.714e-07   ( 7.99%)
  Cell Internal Power  = 1.973e-06   (92.00%)
  Cell Leakage Power   = 4.356e-11   ( 0.00%)
  ---------------------------------------------
Total Power            = 2.145e-06  (100.00%)
-----------------------------------------------------------------------------------------
****************************************
Report : Averaged Power
 ...
****************************************
Current Power Rail: all

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                        Internal   Switching  Leakage    Total
Power Group             Power      Power      Power      Power      (     %)  Attrs
-----------------------------------------------------------------------------------------
io_pad                  0.0000     0.0000     0.0000     0.0000     ( 0.00%)
memory                  0.0000     0.0000     0.0000     0.0000     ( 0.00%)
black_box               0.0000     0.0000     0.0000     0.0000     ( 0.00%)
clock_network           0.0000     0.0000     0.0000     0.0000     ( 0.00%)  i
register                0.0000     0.0000     0.0000     0.0000     ( 0.00%)
combinational           1.842e-05  4.886e-07  9.315e-11  1.891e-05 (100.00%)
sequential              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
-----------------------------------------------------------------------------------------
  Net Switching Power  = 4.886e-07   ( 2.58%)
  Cell Internal Power  = 1.842e-05   (97.42%)
  Cell Leakage Power   = 9.315e-11   ( 0.00%)
 -------------------------------------------------------
Total Power            = 1.891e-05  (100.00%)
-----------------------------------------------------------------------------------------
```

```
1
```

The following example shows a threshold voltage group **report_power** report.

```
pt_shell> report_power -threshold_voltage_group -lvth_groups {LVT}
****************************************
Report : Threshold Voltage Group based leakage power
        -threshold_voltage_group
        -lvth_groups LVT XYZK
 ...
****************************************

Attributes :
------------
         u -> user-defined power group
         L -> user-defined low vth group

Power Group     HVT                 LVT(L)              SVT                 Total
Name            leakage (%)         leakage (%)         leakage (%)         leakage (%)     Attrs
--------------------------------------------------------------------------------------------------------
memory          0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
io_pad          0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
clock_network   0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
black_box       0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
register        5.259e-07 ( 18.86%) 2.263e-06 ( 81.14%)    0.0000 (  0.00%) 2.789e-06 (100.00%)
combinational   0.0000 (  0.00%) 3.223e-06 ( 87.92%) 4.428e-07 ( 12.08%) 3.666e-06 (100.00%)
sequential      0.0000 (  0.00%)    0.0000 (  0.00%) 2.175e-07 (100.00%) 2.175e-07 (100.00%)
--------------------------------------------------------------------------------------------------------
Total           5.259e-07 (  7.88%) 5.486e-06 ( 82.22%) 6.603e-07 (  9.90%) 6.672e-06 (100.00%)
--------------------------------------------------------------------------------------------------------
SUMMARY :
        Low Vth leakage (%)       = 5.486e-06 ( 82.22%)
        Other Vth leakage (%)     = 1.186e-06 ( 17.78%)
        Undefined Vth leakage (%) =    0.0000 (  0.00%)
        ---------------------------------------------
        Total leakage (%)         = 6.672e-06 (100.00%)
--------------------------------------------------------------------------------------------------------

1
```

SEE ALSO

```
create_power_group(2)
current_instance(2)
estimate_clock_network_power(2)
report_power(2)
set_power_analysis_options(2)
update_power(2)
power_clock_network_include_clock_gating_network(3)
power_clock_network_include_register_clock_pin_power(3)
power_enable_leakage_variation_analysis(3)
power_report_leakage_breakdowns(3)
```

report_power_analysis_options

Reports the options for power analysis.

SYNTAX

status **report_power_analysis_options**

ARGUMENTS

None.

DESCRIPTION

The **report_power_analysis_options** command reports the default and user settings for power analysis.

The **collection_result_display_limit** variable specifies the number of cells displayed by the *set_power_analysis_options -cell* command.

EXAMPLES

The following is an example from the **report_power_analysis_options** command output:

```
pt_shell> report_power_analysis_options

****************************************
Report : Power Analysis Options
 ..
****************************************

Power analysis mode : time_based

  Option Name             Value
-------------------------------------
  include                 top
  waveform_format         fsdb
  waveform_output         ptpx
```

SEE ALSO

```
report_power(2)
set_power_analysis_options(2)
update_power(2)
collection_result_display_limit(3)
power_analysis_mode(3)
```

report_power_calculation

Displays the actual calculation of internal power for a pin, leakage power for a cell, or switching power for a net.

SYNTAX

```
integer report_power_calculation
   [-state_condition state]
   [-path_sources name_list]
   [-rise]
   [-fall]
   [-verbose]
   [-report_output prefix]
   [-report_type type]
   [-time]
   object_list
```

Data Types

state	string
name_list	string
prefix	string
type	string
object_list	list

ARGUMENTS

-state_condition *state*

Specifies that the report should be restricted to tables or polynomials with matching state condition. This option can only be specified for objects of type pin or cell. Use the *-state_condition default* option to specify the default state condition. Use the *-state_condition all* option to specify that all states must be considered for reporting.

-path_sources *name_list*

Specifies the related input pin that causes the output pin transition. This option can only be specified for objects of type pin. The path information is used not only for picking the internal power table or polynomial, but also for choosing the input transition time for power calculation if the input transition time is one of the variables. Use the *-path_sources default* option to specify the default path source (without related pin). Use the *-path_sources all* option to specify all possible paths.

-rise

Specifies that the internal power report should be limited to rise transition only. If not specified, both rise and fall transitions are reported, which is equivalent to specifying both the *-rise* and *-fall* options.

-fall

Specifies that the internal power report should be limited to fall transition only. If not specified, both rise and fall

transitions are reported, which is equivalent to specifying both the *-rise* and *-fall* options.

-verbose

Specify this option to increase the amount of information that is reported for cells or pins. For cells, the leakage power calculation report is appended by an internal power calculation report for all pins and all possible states and path sources of the cells in the object list. For pins, all the possible states and paths are printed, which is equivalent to the *-state_condition all -path_sources all* option.

-report_output *prefix*

This option is applicable only in time_based mode and it is specifies the prefix of the reports to be generated. By default the report names are ptpx_detailed.rpt and ptpx_summary.rpt .

-report_type *type*

This option is applicable only in time_based mode and the valid values are 'detailed' , 'summary' and 'both'. If nothing is specified both detailed and summary reports are generated.

-time

Specify the time window for which the report is to be generated in time_based mode. If nothing is specified report is generated for the entire simulation time .

object_list

Specifies the objects for which the power calculation is reported. The type of power calculation depends on the object type. For a pin, internal power calculation is reported. For a cell, a leakage power calculation report is produced. For a net, a switching power calculation report is produced. In time_based mode , only cell object type is allowed.

DESCRIPTION

The **report_power_calculation** command provides detailed power calculation information for the specified pin, cell, or net. You can use this information for debugging or verifying power data in a technology library. Before using this command to display details of internal or leakage power calculation, read the technology library file into the system with the library features attribute, the **report_power_calculation** attribute included in the library - "library_features (report_power_calculation)", which enables the report_power_calculation command for internal, leakage, and net switch- ing power. If the library is missing this feature, then users do not have access to the power table values in library used to calculate power. Therefore, the built-in security mechanism terminates the command. And, details of power calculation information will not be displayed.

In time_based mode this command generates a detailed report on the sequence of events processed for power calculation for the specified list of cells and time window. It also generates a summary report containing the sdpd activites for the specified cells. The options 'state_condition' , 'rise' , 'fall' and 'path_sources' are not applicable in time_based mode.

PrimeTime PX supports power net based power report feature for multivoltage designs. You can use the **set_current_power_net** command to set the power net of interest. Only the power dissipated on the specified power nets is calculated and reported. By default (unless otherwise specified), all the power nets are included in the power analysis. Use the **get_current_power_net** command to show the current power net setting.

EXAMPLES

This example shows the calculation of path dependent internal power from input pin B to output pin Y. The contributions for rise and fall internal power are shown separately.

```
pt_shell> report_power_calculation [get_pin U4/Y] -state default -path B

****************************************
Report : power_calculation
        U4/Y
        -state_condition default
        -path_sources B
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:10:44 2004
****************************************


Library(s) Used:

    example (File: /root/libraries/example.db)


Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example


Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W

Note: Unless specified, the numbers reported are in library units.

================================================================================

Pin Internal Power Calculation
    cell: U4
    pin: Y
    path source: B

  Path Dependent Rise Pin Internal Power = 2.35214e-05 W
    pin internal power = internal energy * pin toggle rate

  Rise internal energy per transition = 0.143423
    library model: NLPM
    table variables:
      X = total_output_net_capacitance = 0.4
      Y = input_net_transition = 0.3
    relevant portion of lookup table:
                     (X)  0.1050      (X)  0.3550
    (Y)  0.0500      (Z)  0.1310      (Z)  0.1410
    (Y)  0.4510      (Z)  0.1320      (Z)  0.1420

       Z = A + B*X + C*Y + D*X*Y
       A =  0.1267           B =  0.0400
```

```
        C =  0.0025           D =  0.0000

      Z = 0.143423
      (no scaling since the library has no internal power k-factors)

    Path Dependent Rise Pin Toggle Rate = 0.164

    Path Dependent Fall Pin Internal Power = 2.35214e-05 W
      pin internal power = internal energy * pin toggle rate

    Fall internal energy per transition = 0.143423
      library model: NLPM
      table variables:
        X = total_output_net_capacitance = 0.4
        Y = input_net_transition = 0.3
      relevant portion of lookup table:
                       (X)  0.1050      (X)  0.3550
      (Y)  0.0500      (Z)  0.1310      (Z)  0.1410
      (Y)  0.4510      (Z)  0.1320      (Z)  0.1420

        Z = A + B*X + C*Y + D*X*Y
        A =  0.1267           B =  0.0400
        C =  0.0025           D =  0.0000

      Z = 0.143423
      (no scaling since the library has no internal power k-factors)

    Path Dependent Fall Pin Toggle Rate = 0.164
```

The following is an example of a state dependent leakage power calculation report.

```
pt_shell> report_power_calculation [get_cell U5] -state "A B"

****************************************
Report : power_calculation
        U5
        -state_condition A B
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:00:48 2004
****************************************

Library(s) Used:

    example (File: /root/libraries/example.db)

Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example


Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W
```

```
Note: Unless specified, the numbers reported are in library units.

================================================================================
Cell Leakage Power Calculation
    cell: U5
    state condition: A B

  State Dependent Leakage Power = 3.184e-12 W
    cell leakage power = leakage power value * state probability

    Leakage Power Value = 1.021e-07
      library model: scalar
      value = 1.021e-07
      (no scaling since the library has no leakage power k-factors)

    State Probability = 0.03120   (estimated)
```

The following shows how switching power calculation is performed.

```
pt_shell> report_power_calculation [get_net q]

****************************************
Report : power_calculation
         q
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:12:32 2004
****************************************


Library(s) Used:

    example (File: /root/libraries/example.db)

Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example

Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W

Note: Unless specified, the numbers reported are in library units.

================================================================================

Net Switching Power Calculation
    net: q
    driver: U4/Y

  Switching power = 1.296e-04 W
    net switching power = switching energy * net toggle rate

  Switching Energy Per Transition = 0.162
    switching energy = 0.5 * capacitance * voltage ^ 2
    total net capacitance = 0.4
```

```
        voltage = 0.9

      Net Toggle Rate = 0.8 (user annotated)
```

SEE ALSO

```
read_saif(2)
report_lib(2)
report_power(2)
set_current_power_net(2)
set_switching_activity(2)
update_power(2)
power_analysis_mode(3)
```

report_power_derate

Reports power derate factors annotated on the current design.

SYNTAX

```
int report_power_derate
   [-include_inherited]
   [-significant_digits digits]
   [-nosplit]
   object_list
```

Data Types

object_list	list
digits	int

ARGUMENTS

-include_inherited

Indicates that the power derating factors reported on each object should also include those set by other objects in the design.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for values in the report. Allowed values are 0-12; the default is determined by the report_default_significant_digits variable. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The -nosplit option prevents line-splitting and facilitates writing software to extract information from the report output.

object_list

Specifies current design or a list of cells or library cells from which the power derating factors are reported.

DESCRIPTION

This command is used to report the derate factors either on the design, specific cells or library cells contained in the design. If this command is called without any option, a separate report will be invoked for the power derating

factors stored on the current design (referred to as global power derating factors) and also for each cell or library cell that has power derating factors set on it.

If this command is called with the boolean option *-include_inherited*, each row containing an inherited derating factor will contain a column entry indicating the object from which it was inherited. If the command is called with the *object_list* specified then derate reports will only be issued for the instances specified in the object list.

For significant digits used in power derate report, the *-significant_digits* option can be used. If the option is not specified, the value of the *report_default_significant_digits* variable is applied. By default, the number of significant digits used to display values is 2.

EXAMPLES

In the following example, power derating factors have been set globally on the design; therefore, only global derate factors are reported.

```
pt_shell> set_power_derate -switching 0.92
pt_shell> set_power_derate -internal  1.15
pt_shell> set_power_derate -gate_leakage 0.75
pt_shell> set_power_derate -subthreshold 1.26
pt_shell> report_power_derate

pt_shell> report_power_derate
****************************************
Report : power derate
Design : top
Version: D-2009.12
Date   : Mon Sep 25 11:08:59 2009
****************************************

                                                                                  Subthreshold    Gate
    Object Name    Object Type           Switching   Internal    Leakage      Leakage         Leakage
 ---------------------------------------------------------------------------------------------------
    top            design                0.92        1.15        --           1.26            0.75
1
```

In the following example power derating factors have been set on the design named 'top'. More restrictive derates have been set on the hierarchical block name 'H1'. The derates that apply to the hierarchical cell named, H1/U1, are reported.

```
pt_shell> set_power_derate 0.95
pt_shell> set_power_derate -switching  1.06
pt_shell> set_power_derate -internal -gate_leakage 0.82 [get_cell H1]
pt_shell> set_power_derate -subthreshold_leakage 1.12 [get_cell H1/U1]
pt_shell> report_power_derate -include_inherited [get_cells H1/U1]

****************************************
Report : power derate
        -include_inherited
Design : top
Version: D-2009.12
Date   : Mon Sep 25 11:13:40 2009
****************************************

                                                                                  Subthreshold    Gate
  Object Name    Object Type           Switching   Internal    Leakage      Leakage         Leakage
---------------------------------------------------------------------------------------------------
  H1/U1          cell (leaf)           1.06        0.82        --           1.12            0.82
  Inherited                            top         H1          H1           --              H1
```

```
1
```

SEE ALSO

```
reset_power_derate(2)
set_power_derate(2)
```

report_power_domain

Reports the specified power domain.

SYNTAX

```
integer report_power_domain
   [domain_list]
```

Data Types

```
domain_list          list
```

ARGUMENTS

domain_list

Specifies the list of power domains reported. The names in the list should be a simple (nonhierarchical) name.

If there is no such a power domain with the same name in the current scope, the command fails.

If this option is not specified, all power domain in the specified scope is reported.

DESCRIPTION

The **report_power_domain** command enables you to get the detailed information of an existing power domain in the current scope. The information of the power domain includes the full name, scope, element, and primary power and ground net.

The command returns 1 if it succeeds, and returns 0 otherwise.

EXAMPLES

Here are examples of the **report_power_domain** output:

```
pt_shell> report_power_domain
****************************************
Report : power domains
Design : top
Version: A-2007.12
```

```
Date   : Thu Sep 27 18:59:04 2007
****************************************

  Total of 2 power domains defined for design 'top'.

    Power Domain : top_domain
    Scope : top level
    Elements : top level

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>
  ---------------------------------------------------------------------------

    Power Domain : PD0
    Scope : <top level>
    Elements :  PD0_INST

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>
  ---------------------------------------------------------------------------

1

pt_shell> report_power_domain [get_power_domain top_domain]
****************************************
Report : power domains
Design : top
Version: A-2007.12-Beta2-DEV
Date   : Thu Sep 27 18:59:04 2007
****************************************

    Power Domain : top_domain
    Scope : <top level>
    Elements : <top level>

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>
  ---------------------------------------------------------------------------

1

pt_shell> report_power_domain [get_power_domain -regexp {t.*}]
****************************************
Report : power domains
Design : top
Version: A-2007.12
Date   : Thu Sep 27 18:59:04 2007
****************************************

    Power Domain : top_domain
    Scope : <top level>
    Elements : <top level>

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>
  ---------------------------------------------------------------------------

1
```

SEE ALSO

```
create_power_domain(2)
get_power_domains(2)
```

report_power_groups

Report the existing power groups.

SYNTAX

```
string report_power_groups
   [group_names]
   [-nosplit]
```

Data Types

```
group_names        list
```

ARGUMENTS

group_names

Specifies the power group names that are to be reported.

-nosplit

Indicates to prevent line splitting if column overflows.

DESCRIPTION

The **report_power_groups** command reports the power groups that has been created in the earlier stage, including the predefined power groups.

EXAMPLES

In the following example, all the existing power groups are reported.

```
pt_shell> create_power_group -name clock_tree [get_clock_network_objects -type cell]
1
pt_shell> report_power_groups

i****************************************
Report : report_power_groups
        -nosplit
Design : testcase
```

```
Version: Y-2006.06-Beta1-DEV
Date   : Mon Mar 13 16:28:48 2006
****************************************

  Power Group                          Size           Attribute
-------------------------------------------------------------------
black_box                                 0       Default
clock_network                             5       Default
combinational                             0       Default
io_pad                                    0       Default
memory                                    0       Default
register                                 25       Default
sequential                                0       Default
clock_tree                                5       User defined

1
```

SEE ALSO

```
create_power_group(2)
remove_power_groups(2)
get_power_group_objects(2)
```

report_power_pin_info

Reports the power pin information for logic library cells or leaf cells.

SYNTAX

```
status report_power_pin_info
   object_list
```

Data Types

```
object_list      list
```

ARGUMENTS

object_list

Specifies a list of logic library cells or a list of leaf cells or ports.

DESCRIPTION

The **report_power_pin_info** command reports the power pin information for logic library cells or leaf level cells in the current design.

If you specify a list of library cells, the name, the types, and the voltage specification of the power pins are reported. All types of PG pins including power, ground, and bias pins are shown.

If you specify a list of leaf cells as the *object_list*, the power net information that are connected to the power pins are also reported. Hierarchical cells are ignored by this command.

EXAMPLES

The following example uses the command to report the power pin information.

```
pt_shell> report_power_pin_info [get_cells -hierarchical]

****************************************
Report : power pin info
Design : top
 ...
```

```
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name      Type                Voltage   Power Net Connected
-----------------------------------------------------------------------------------------
PD0_INST/I0     PWR                 primary_power        1.0000
PD0_INST/I0     GND                 primary_ground       0.0000
PD0_INST/I1     PWR                 primary_power        1.0000
PD0_INST/I1     GND                 primary_ground       0.0000
I0              PWR                 primary_power        1.0000
I0              GND                 primary_ground       0.0000
I1              PWR                 primary_power        1.0000
I1              GND                 primary_ground       0.0000
1

pt_shell> report_power_pin_info [get_lib_cells -of_objects [get_cells -hierarchical]]

****************************************
Report : power pin info
Design : top
 ...
****************************************

Library Cell    Power Pin Name      Type                Voltage
-------------------------------------------------------------
INVX2           PWR                 primary_power        1.0000
INVX2           GND                 primary_ground       0.0000
BUFX2           PWR                 primary_power        1.0000
BUFX2           GND                 primary_ground       0.0000
AND2X1          PWR                 primary_power        1.0000
AND2X1          GND                 primary_ground       0.0000
1
pt_shell> connect_power_net_info I0 -power_pin_name PWR -power_net_name exp_VDD
1
pt_shell> connect_power_net_info PD0_INST/I0 -power_pin_name PWR -power_net_name exp_VDD
1
pt_shell> report_power_pin_info [get_cells -hierarchical]

****************************************
Report : power pin info
Design : top
 ...
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name      Type                Voltage   Power Net Connected
-----------------------------------------------------------------------------------------
PD0_INST/I0     PWR                 primary_power        1.0000   exp_VDD (*)
PD0_INST/I0     GND                 primary_ground       0.0000   A_VSS
PD0_INST/I1     PWR                 primary_power        1.0000   A_VDD
PD0_INST/I1     GND                 primary_ground       0.0000   A_VSS
I0              PWR                 primary_power        1.0000   exp_VDD (*)
I0              GND                 primary_ground       0.0000   T_VSS
I1              PWR                 primary_power        1.0000   T_VDD
I1              GND                 primary_ground       0.0000   T_VSS
1

pt_shell> report_power_pin_info [get_cells -hierarchical]

****************************************
Report : power pin info
Design : top
```

```
 ...
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name      Type                Voltage   Power Net Connected
-------------------------------------------------------------------------------------
PD0_INST/I0     PWR                 primary_power        1.0000   exp_VDD (*)
PD0_INST/I0     GND                 primary_ground       0.0000   A_VSS
PD0_INST/I1     PWR                 primary_power        1.0000   A_VDD
PD0_INST/I1     GND                 primary_ground       0.0000   A_VSS
I0              PWR                 primary_power        1.0000   exp_VDD (*)
I0              GND                 primary_ground       0.0000   T_VSS
I1              PWR                 primary_power        1.0000   T_VDD
I1              GND                 primary_ground       0.0000   T_VSS
1
```

SEE ALSO

connect_power_domain(2)
connect_power_net_info(2)

report_power_rail_mapping

Reports the current power rail mapping.

SYNTAX

```
status report_power_rail_mapping
   [-cells cell_list]
   [-verbose]
   [-nosplit]
```

Data Types

cell_list	list

ARGUMENTS

-cells *cell_list*

Specifies the instance names or collections of instances. This only takes effect when the **-verbose** option is used. The instance can be hierarchical or leaf instances. For a instance block, only the top instance needs to be specified. All the leaf instances inside such blocks are reported. Without this option, the mapping is reported for the whole design.

-verbose

Reports detailed rail mapping for each individual leaf instance. Choose the leaf instances to be reported by using the **-instance** option. When this option is used, PrimeTime PX also checks potential problems in rail mapping done for the instances. The list of checks are:

- Different library rails are mapped to the same design rail in a multirail leaf cell
- No rail specific power table is defined for a library cell with multiple power rails

-nosplit

Most of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

The **report_power_rail_mapping** command displays all the power rails defined in the library and those existing

in the design. The library power rails are the *voltage_map* attributes defined in the library (or the *power_rail* attributes defined in the **power_supply** group). The design power rails are the rails created by the **create_power_rail_mapping** command. If a library does not have rail information, default_rail is shown for its library power rail. If the mapping does not exist, the default design power rail is shown and labeled with *(default)*. During power simulation, those missing cases are actually mapped to the default design power rail. Internally, there is a default rail associated with each library and with the whole design. If the **-verbose** option is used, PrimeTime PX reports the detailed rail mapping for each individual leaf instances and also shows possible rail mapping problems for design.

EXAMPLES

Here is an example of the **report_power_rail_mapping** command:

```
pt_shell> report_power_rail_mapping -verbose -cells block1
```

SEE ALSO

create_power_rail_mapping(2)

report_power_switch

Report all power switches defined in the design.

SYNTAX

```
int report_power_switch
```

ARGUMENTS

None.

DESCRIPTION

The **report_power_switch** command enables you to get the detail information of all power_switches defined in the design. The power switch information includes:

- full name
- power domain where it is created in
- output supply port
- input supply port
- control port
- on_state statement

The command returns 1 if it succeed; otherwise, it returns 0.

EXAMPLES

The following example reports the information of all power_switches defined:

```
pt_shell> report_power_switch
```

```
  Power Switch : SW1
----------------------------------------
    Power Domain : PD1
    Output Supply Port : vout VN2
    Input Supply Port : vin VN1
    Control Port : ctrl_small ON1
    Control Port : ctrl_large ON2
    On State : state1 vin { ON1 & ON2 }
----------------------------------------
1
```

SEE ALSO

create_power_switch(2)

report_probes

Reports pg_pins with voltage and current probes.

SYNTAX

```
string report_probes
   [pg_pins]
```

Data Types

```
list pg_pins
```

ARGUMENTS

pg_pins

Specifies a list of pg_pins to be inspected for reporting. If not specified, all pg_pins in the design are inspected.

DESCRIPTION

The **report_probes** command reports a list of pg_pins with voltage and current probes. If *pg_pins* argument is specified, only those pg_pins in the specified list are inspected and reported. Otherwise, all pg_pins in the design are inspected and those with probes are reported. When *-fsdb_file* option is used in **calculate_rail_voltage** command, both voltage and current waveforms at the probed pg_pins are saved in the FSDB file during dynamic analysis.

EXAMPLES

The following example shows how to report pg_pins with voltage and current probes.

```
pr_shell> report_probes
****************************************
Report : pg_pins with voltage & current probes
Design : design
Version: I-2013.12-DEV
Date   : Tue Dec 17 14:46:37 2013
****************************************

INV21/vdd
```

```
INV21/vss
INV61/vdd
INV61/vss
INV22/vdd
INV22/vss
INV62/vdd
INV62/vss
INV11/vdd
INV11/vss
INV51/vdd
INV51/vss
INV12/vdd
INV12/vss
INV52/vdd
INV52/vss
INV41/vdd
INV41/vss
INV42/vdd
INV42/vss
INV31/vdd
INV31/vss
INV32/vdd
INV32/vss
1
```

SEE ALSO

```
set_probes(2)
remove_probes(2)
calculate_rail_voltage(2)
```

report_pulse_clock_max_transition

Displays maximum transition computation at the input of pulse generator and pulse clock network.

SYNTAX

```
int report_pulse_clock_max_transition
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-rise]
   [-fall]
   [-transitive_fanout]
   [port_pin_list]
```

Data Types

digits	int
port_pin_list	list

ARGUMENTS

-all_violators

Specifies that only the pins violating maximum transition constraint are reported.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

-rise

Report slack for rise transition.

-fall

Report slack for fall transition.

-transitive_fanout

Report transition slack at the transitive fanout of pulse generator cells. If this option is not set, the transition

slack is reported at the input of pulse generator cells.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock maximum transition constraint.

DESCRIPTION

The **report_pulse_clock_max_transition** command displays the maximum transition computation of pins and ports. By default, only the input pins of the pulse generators are reported. If rise or fall is not specified, both rise and fall transition slack are reported. To enable pulse clock constraint checking, ensure the **timing_enable_pulse_clock_constraints** variable is set to *true*, which is its default.

EXAMPLES

The following example generates a report of all transition slack computation at the input of pulse generators in the current design.

```
pt_shell> report_pulse_clock_max_transition

****************************************
Report : pulse clock max transition
        -rise
        -fall
Design : test
****************************************

   pulse_clock_max_transition_rise

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ---------------------------------------------------------------
   p1/i                        0.40           0.12           0.28  (MET)

   pulse_clock_max_transition_fall

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ---------------------------------------------------------------
   p1/i                        0.40           0.12           0.28  (MET)
```

The following example generates a report of all transition slack computation in the pulse clock network in the current design.

```
pt_shell> report_pulse_clock_max_transition -transitive_fanout

****************************************
Report : pulse clock max transition
        -rise
        -fall
Design : test
****************************************
```

```
pulse_clock_max_transition_rise_transitive_fanout

                          Required        Actual
Pin                      Transition     Transition        Slack
-------------------------------------------------------------------
ff2/cp                       0.20           0.08           0.12  (MET)
ff1/cp                       0.20           0.02           0.18  (MET)
u3/zn                        0.30           0.08           0.22  (MET)
p1/z                         0.30           0.02           0.28  (MET)
u3/i                         0.30           0.02           0.28  (MET)

pulse_clock_max_transition_fall_transitive_fanout

                          Required        Actual
Pin                      Transition     Transition        Slack
-------------------------------------------------------------------
ff2/cp                       0.20           0.08           0.12  (MET)
ff1/cp                       0.20           0.02           0.18  (MET)
u3/zn                        0.30           0.08           0.22  (MET)
p1/z                         0.30           0.02           0.28  (MET)
u3/i                         0.30           0.02           0.28  (MET)
```

SEE ALSO

set_pulse_clock_max_transition(2)
remove_pulse_clock_max_transition(2)
report_constraint(2)

report_pulse_clock_max_width

Displays maximum pulse width computation for the pulse clock network.

SYNTAX

```
int report_pulse_clock_max_width
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-path_type format]
   [-transitive_fanout]
   [port_pin_list]
```

Data Types

digits	int
format	string
port_pin_list	list

ARGUMENTS

-all_violators

Specifies that only the pins in the pulse clock network violating the maximum pulse width constraint are reported.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

-path_type *format*

Specifies the format of the path report and how the clock path is displayed. The allowed values are its default of *summary* and also *short*. The *summary* value generates a report with a column format that shows one line for each path and shows only the required pulse width, actual pulse width and slack; The *short* value displays only startpoints and endpoints in the clock path.

-transitive_fanout

Report constraints at the transitive fanout of pulse generators. Currently, specifying or not specifying this option has the same behavior.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock maximum width constraint.

DESCRIPTION

The **report_pulse_clock_max_width** command displays the maximum pulse clock width computation of the specified pins and ports. By default all pins with pulse clock width constraint are reported. To enable pulse clock constraint checking, ensure that the **timing_enable_pulse_clock_constraints** variable is set to *true*, which is the default.

EXAMPLES

The following example generates a report of all maximum pulse width violations in the current design.

```
pt_shell> report_pulse_clock_max_width

****************************************
Report : pulse clock max width
        -path_type summary
Design : test
****************************************

   sequential_pulse_clock_max_width

                       Required       Actual
   Pin                 pulse width    pulse width       Slack
   ----------------------------------------------------------------
   ff1/cp (high)           1.00           0.54          0.46  (MET)
   ff2/cp (low)            1.00           0.56          0.44  (MET)


   clock_tree_pulse_clock_max_width

                       Required       Actual
   Pin                 pulse width    pulse width       Slack
   ----------------------------------------------------------------
   u3/zn (low)             1.00           0.56          0.44  (MET)
   p1/z (high)             1.00           0.54          0.46  (MET)
   u3/i (high)             1.00           0.54          0.46  (MET)
```

SEE ALSO

```
remove_pulse_clock_max_width(2)
report_constraint(2)
set_pulse_clock_max_width(2)
```

```
report_default_significant_digits(3)
timing_enable_pulse_clock_constraints(3)
```

report_pulse_clock_min_transition

Displays minimum transition computation at the input of pulse generator.

SYNTAX

```
int report_pulse_clock_min_transition
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-rise]
   [-fall]
   [port_pin_list]
```

Data Types

```
digits            int
port_pin_list     list
```

ARGUMENTS

-all_violators

Specifies that only the input pins of pulse generators violating minimum transition constraint is reported.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which defaults to 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

-rise

Reports slack for rise transition.

-fall

Reports slack for fall transition.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock minimum transition constraint.

DESCRIPTION

The **report_pulse_clock_min_transition** command displays the minimum transition computation of pins and ports. By default, all pins with pulse clock minimum transition constraint are reported. Similarly, if rise or fall is not specified, then both rise and fall transition slack is reported. To enable pulse clock constraint checking, ensure the **timing_enable_pulse_clock_constraints** variable is set to its default of *true*.

EXAMPLES

The following example generates a report of all minimum transition violations in the current design.

```
pt_shell> report_pulse_clock_min_transition
report_pulse_clock_min_transition
****************************************
Report : pulse clock min transition
        -rise
        -fall
Design : test
****************************************


   pulse_clock_min_transition_rise

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ---------------------------------------------------------------
   p1/i                        0.30           0.12          -0.18  (VIOLATED)

   pulse_clock_min_transition_fall

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ---------------------------------------------------------------
   p1/i                        0.30           0.12          -0.18  (VIOLATED)
```

SEE ALSO

```
set_pulse_clock_min_transition(2)
remove_pulse_clock_min_transition(2)
report_constraint(2)
```

report_pulse_clock_min_width

Displays minimum pulse width computation for the pulse clock network.

SYNTAX

```
int report_pulse_clock_min_width
   [-all_violators]
   [-significant_digits digits]
   [-nosplit]
   [-path_type format]
   [-transitive_fanout]
   [port_pin_list]
```

Data Types

digits	int
format	string
port_pin_list	list

ARGUMENTS

-all_violators

Specifies that only the pins in the pulse clock network violating the minimum pulse width constraint is reported.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

-path_type *format*

Specifies the format of the path report and how the clock path is displayed. Allowed values are *summary* (the default) and *short*. The *summary* value generates a report with a column format that shows one line for each path and shows only the required pulse width, actual pulse width, and slack. The *short* value displays only start and end points in the clock path.

-transitive_fanout

Reports constraints at the transitive fanout of pulse generators. In this release, specifying or not specifying this

option has the same behavior.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock minimum width constraint.

DESCRIPTION

The **report_pulse_clock_min_width** command displays the minimum pulse clock width computation of the specified pins and ports. By default, all pins with pulse clock width constraint are reported. To enable pulse clock constraint checking, ensure the *timing_enable_pulse_clock_constraints* variable is set to its default of *true*.

EXAMPLES

The following example generates a report of all minimum pulse width violations in the current design.

```
pt_shell> report_pulse_clock_min_width
****************************************
Report : pulse clock min width
         -path_type summary
Design : test
****************************************


   sequential_pulse_clock_min_width

                            Required        Actual
   Pin                     pulse width    pulse width       Slack
   ---------------------------------------------------------------
   ff2/cp (low)                2.00           0.56          -1.44  (VIOLATED)
   ff1/cp (high)               0.80           0.54          -0.26  (VIOLATED)


   clock_tree_pulse_clock_min_width

                            Required        Actual
   Pin                     pulse width    pulse width       Slack
   ---------------------------------------------------------------
   u3/i (high)                 0.80           0.54          -0.26  (VIOLATED)
   u3/zn (low)                 0.80           0.56          -0.24  (VIOLATED)
   p1/z (high)                 0.80           0.54          -0.26  (VIOLATED)
```

SEE ALSO

```
set_pulse_clock_min_width(2)
remove_pulse_clock_min_width(2)
report_constraint(2)
```

report_qor

Provides an overview of the quality of a design.

SYNTAX

```
status report_qor
   [-max]
   [-min]
   [-significant_digits digits]
   [-only_violated]
   [-summary]
   [-physical]
   [-pba_mode none | path | exhaustive]
```

Data Types

```
digits     integer
```

ARGUMENTS

-max

Indicates that only maximum delay and setup information is to be displayed. If this option is not specified, the report for both min and max delay types is produced.

-min

Indicates that only minimum delay and hold information is to be displayed. If this option is not specified, the report for both min and max delay types is produced.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which has a default of 2. Use this option to override the default.

-only_violated

Shows timing path group summaries for only violating path groups.

-summary

Summarizes the worst negative slack (WNS) and total negative slack (TNS) for all path groups. See **timing_report_union_tns** for details of TNS calculation.

-physical

This option has no effect in PrimeTime.

-pba_mode none | path | exhaustive

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.

DESCRIPTION

The **report_qor** command provides an overview of the quality of results (QoR) for a design, including the length of the worst paths (shown as "Critical Path Length" in the report, which represents the arrival time of the signal at the path's endpoint), total negative slack, minimum delay and hold summaries, and a detailed DRC summary.

None of the options are supported at the Master in distributed analysis mode. All the options are supported with the command, if executed on remote slaves as part of the distributed command **remote_execute -partitions**.

If you specify the **-pba_mode exhaustive** option, path-based analysis is used to calculate the slack values. This option requires that a path search be performed to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases.

EXAMPLES

The following example shows detailed QoR information for the current design; some timing path group summaries are omitted for clarity:

```
pt_shell> report_qor

****************************************
Report : qor
Design : example
 ...
****************************************

Timing Path Group '**async_default**' (max_delay/setup)
-----------------------------------------
Levels of Logic:                          50
Critical Path Length:                  20.42
Critical Path Slack:                  -23.71
Total Negative Slack:               -1010.10
No. of Violating Paths:                  186
-----------------------------------------

Timing Path Group 'ADC_SCLK_IN' (max_delay/setup)
-----------------------------------------
Levels of Logic:                          20
Critical Path Length:                   6.31
Critical Path Slack:                   -2.10
```

```
Total Negative Slack:                  -39.80
No. of Violating Paths:                    32
----------------------------------------------

Timing Path Group 'CLK1' (max_delay/setup)
----------------------------------------------
Levels of Logic:                            7
Critical Path Length:                    2.62
Critical Path Slack:                     0.26
Total Negative Slack:                    0.00
No. of Violating Paths:                     0
----------------------------------------------

Timing Path Group '**async_default**' (min_delay/hold)
----------------------------------------------
Levels of Logic:                            3
Critical Path Length:                    0.79
Critical Path Slack:                    -3.37
Total Negative Slack:                  -13.19
No. of Violating Paths:                     4
----------------------------------------------

Timing Path Group 'ADC_PCMCLK_IN' (min_delay/hold)
----------------------------------------------
Levels of Logic:                            2
Critical Path Length:                    0.84
Critical Path Slack:                    -1.54
Total Negative Slack:                  -23.26
No. of Violating Paths:                    33
----------------------------------------------

Timing Path Group 'C_MAU/KDmem_ctl0/st20_reg_q_regx3x/Q' (min_delay/hold)
----------------------------------------------
Levels of Logic:                            1
Critical Path Length:                    0.42
Critical Path Slack:                     0.38
Total Negative Slack:                    0.00
No. of Violating Paths:                     0
----------------------------------------------

Cell Count
----------------------------------------------
Hierarchical Cell Count:                 2310
Hierarchical Port Count:                38125
Leaf Cell Count:                        24230
----------------------------------------------

Area
----------------------------------------------
Design Area:                       5659902.00
----------------------------------------------

Design Rule Violations
----------------------------------------------
Total No. of Pins in Design:            87497
max_capacitance Count:                    305
max_fanout Count:                         105
clock_gating_setup Count:                  32
clock_gating_hold Count:                   37
max_capacitance Cost:                  -52.87
max_fanout Cost:                     -3202.00
clock_gating_setup Cost:              -409.27
clock_gating_hold Cost:               -138.85
Total DRC Cost:                      -3802.99
```

```
-----------------------------------------------
```

SEE ALSO

```
report_analysis_coverage(2)
report_constraint(2)
report_global_timing(2)
report_default_significant_digits(3)
timing_report_union_tns(3)
```

report_qtm_model

Reports quick timing model (QTM) data.

SYNTAX

```
string report_qtm_model
   [-global_parameters]
   [-ports]
   [-arcs]
   [-nosplit]
```

ARGUMENTS

-global_parameters

Reports the global parameters of the current quick timing model.

-ports

Reports the ports of the current quick timing model.

-arcs

Reports the timing arcs of the current quick timing model.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

DESCRIPTION

The **report_qtm_model** command prints a report of the current quick timing model. By default, the command reports the global parameters, ports, and arcs of the model.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command reports all the attributes of the currently active quick timing model.

```
pt_shell> report_qtm_model

****************************************
Report : qtm_model
 ...
****************************************

PATH TYPES
Type     Cell      Input pin   Output pin   Fanout           Delay per level
-------------------------------------------------------------------------------
path1    AN210     A           Y            2                0.570929

DRIVE TYPES
Type     Cell      Input pin   Output pin
-------------------------------------------------------------------------------
drive1   AN210     A           Y

LOAD TYPES
Type     Cell      Input pin      Capacitance
-------------------------------------------------------------------------------
load1    AN210     A              1

CLOCK RELATED PARAMETERS
Parameter    Cell      Clock Pin       Related Pin    Delay
-------------------------------------------------------------------------------
setup        DTN10     CLK             D              1.33
hold         DTN10     CLK             D              1.33
setup        DTN10     CLK             Q              1.80244

***********************************************
Wire load model not defined
***********************************************
***********************************************
Max Transition not defined
***********************************************
PORTS
Port     Direction Width Cap Value  Load Type  Drive Value  Drive Type
-------------------------------------------------------------------------------
CLK      CLOCK     1      -           -         -            -
A        INPUT     1      2         load1       -            -
B        INPUT     1      2         load1       -            -
X        OUTPUT    1      -           -         -           drive1
Y        OUTPUT    1      -           -         -           drive1

ARCS
Name     From                To                  Delay  Path Type   Path Factor
-------------------------------------------------------------------------------
CLK_to_A CLK                 A                   1.142  path1        2
CLK_to_B CLK                 B                   3.997  path1        7
CLK_to_X CLK                 X                   3.000   -           -
CLK_to_Y CLK                 Y                   7.000   -           -
```

The following command reports the global parameters of the currently active quick timing model.

```
pt_shell> report_qtm_model -global_parameters

****************************************
Report : qtm_model
 ...
****************************************

PATH TYPES
```

```
Type     Cell      Input pin   Output pin   Fanout           Delay per level
-------------------------------------------------------------------------------
path1    AN210     A           Y            2                0.570929

DRIVE TYPES
Type     Cell      Input pin   Output pin
-------------------------------------------------------------------------------
drive1   AN210     A           Y

LOAD TYPES
Type     Cell      Input pin      Capacitance
-------------------------------------------------------------------------------
load1    AN210     A              1

CLOCK RELATED PARAMETERS
Parameter    Cell      Clock Pin       Related Pin    Delay
-------------------------------------------------------------------------------
setup        DTN10     CLK             D              1.33
hold         DTN10     CLK             D              1.33
setup        DTN10     CLK             Q              1.80244

**********************************************
Wire load model not defined
**********************************************
**********************************************
Max Transition not defined
**********************************************
```

SEE ALSO

create_qtm_model(2)
save_qtm_model(2)

report_rail_checks

Reports the accumulated status of the rail analysis readiness checks that are performed on the design during multiple steps of the current session.

SYNTAX

status **report_rail_checks**

ARGUMENTS

None

DESCRIPTION

The command provides a summary report of all of the rail analysis readiness checks performed on the design and related data during the analysis session. These readiness checks are performed to determine whether there are problems with the design or library setup that could potentially invalidate the results of the analysis you perform. The readiness report documents many issues, including the following:

- Unrealistically large intrinsic RC values on a PG pin
- Unrealistically large power values derived for a cell
- Supply net shapes that are not connected to any ideal voltage sources (that is, tap points)

By default, PrimeRail performs certain design readiness checks whenever the data is needed or when setting up the design or preparing data for analysis. For example, PrimeRail performs readiness checks when executing the **open_mw_cel**, **read_supply_parasitics**, **update_currents**, and **calculate_rail_voltage** commands. A summary message is issued whenever a check is performed or a problem is found. PrimeRail does not issue a message when no problem is found. The notification message includes a brief description of the problem, the number of times that the problem is found, and a pointer to the file that contains details about each specific problem.

For example:

pr_shell> open_mw_cel -lib MyLib MyDesign

... (information) For 15 library master cells the number of PG pins of the related reference cells in the Liberty (.db) file is not the same as the number of PG pins in the Milkyway physical view. Refer to the file ./PR_CHECKING_DIR/lib.inconsistent_library_pg_pins for details. (MKW-017)

PrimeRail writes all issues identified by the rail readiness checks to log files which are saved in the UNIX working directory. The **pr_checking_dir** TCL variable contains the path to the checking directory. The checking directory

remains after the session is terminated, allowing you to review the checking files after your run completes. If you start a new session and a checking directory already exists in the working directory, the new directory is created with an additional suffix (like _1, _2, and so on). If the suffix you assign already beeing used in the existing checking directory, the command deletes the existing checking directory to prevent from having log files from multiple sessions in the same checking directory. If you want to ensure that the checking logs are not deleted by a later session, copy the directory to another location before the session is terminated.

The report generated by the **report_rail_checks** command includes the following information for each readiness check:

- A brief description of the check
- The ID number of the summary message generated when the check identifies an issue. See the message man page, in the form of man <ID>, to get more detailed information on the specific readiness check, including how to resolve the identified issues.
- A "Yes/No" flag indicating whether the check has been performed in this session
- A count of the number of individual issues identified by the check, if it has been performed

EXAMPLES

The following example runs the **report_rail_checks** command to generate a readiness report immediately after the design has been loaded.

```
pr_shell> open_mw_cel -lib MyLib MyDesign
...
pr_shell> report_rail_checks

****************************************
Report : report_rail_checks
Design : MyDesign
Version: E-2010.12
Date   : Wed Dec  8 09:47:40 2010
****************************************

Validity Check                        Manpage ID     Checked     Issues
-----------------------------------------------------------------------
Missing library PG pins               RAIL-001       Yes              0
Mis-matched supply PG pins            PGEX-003       No               -
More Milkyway PG pins than library    MKW-019        Yes              0
Huge peak current                     RAIL-109       No               -
Inconsistent Milkyway library PG pins MKW-017        Yes              0
Physically floating PG pins           RAIL-301       No               -
Unmatched Milkyway library PG pins    MKW-018        Yes              5
Physically unconnected PG pins        RAIL-010       No               -
Missing power-arc CCSP waveform       RAIL-110       No               -
Under interpolation                   RAIL-112       No               -
Huge intrinsic rc                     RAIL-114       No               -
Missing library macro PG pins         PGEX-004       No               -
Logically floating PG nets            RAIL-116       Yes             23
Extracted terminals with no PG pins   PGEX-002       No               -
Huge capacitance or slew              RAIL-108       No               -
Milkyway derived PG pins              MKW-016        Yes              0
Missing library CCSP waveform         RAIL-002       Yes              1
Over extrapolation                    RAIL-111       No               -
Large power value                     RAIL-113       No               -
Logically floating cells              RAIL-115       Yes              0
```

```
Extracted unconnected terminals       PGEX-005       No               -
1
```

SEE ALSO

calculate_rail_voltage(2)
open_mw_cel(2)
pr_checking_dir(3)
pr_work_dir(3)
read_supply_parasitics(2)
update_currents(2)

report_rail_current

Reports the current that has been set by the **set_rail_current** command.

SYNTAX

```
status report_rail_current
   -lib_cell library_cell
   [-pg_pin pg_pin]
```

Data Types

```
list library_cell
string pg_pin
```

ARGUMENTS

-lib_cell *library_cell*

Specifies a list library master cells to report user set currents.

-pg_pin *pg_pin*

Specifies a PG pin name on which user set rail current is reported. When this option is not specified, user set rail currents on all PG pins will be reported.

DESCRIPTION

This command provides a report of the currents that have been set by the **set_rail_current** command through the option *-lib_cell*. The command returns 1 if successful, and 0 if unsuccessful.

This command can only be used before the design is linked through the command **open_mw_cel** or **open_block**.

EXAMPLES

The following example shows a sample report for all the PG pins with user set currents, where the currents are set from **set_rail_current command.**

```
pr_shell> set_rail_current -lib_cell buf64 -pg_pin VDD -waveform {0.0 0.0 3.0e-9 -0.003 6.0e-9 -0.003
pr_shell> set_rail_current -lib_cell buf64 -pg_pin VSS -waveform {0.0 0.0 3.0e-9 0.003 6.0e-9 0.003 9
pr_shell> report_rail_current -lib_cell buf64

****************************************
Report : User set rail current
Design :
****************************************

time unit: ns
current unit: mA

Library cell: buf64
PG pin VDD user set PWL current: ( 0 0 ) ( 3 -3 ) ( 6 -3 ) ( 9 0 )
PG pin VSS user set PWL current: ( 0 0 ) ( 3 3 ) ( 6 3 ) ( 9 0 )
```

SEE ALSO

set_rail_current(2)

report_rail_macro_model

Reports the information for rail macro models that are created by the **create_rail_macro_model** command.

SYNTAX

```
status report_rail_macro_model
   [-library library_name]
   [-cell cell_name]
   [-lib_cell lib_cell_list]
   [-macro_name macro_name_list]
   [-loaded_model_list]
   [-unmodeled_macro_list]
   [-include_unmodeled]
   [-report_instances]
   [-pin_detail]
   [-internal_net_detail]
   [-mode_detail]
   [-dump_mode_waveform]
   [-waveform_filename file_name]
   [-waveform_filetype fsdb|text]
```

Data Types

```
string library_name
string cell_name
string file_name
list lib_cell_list object
list macro_name_list string
```

ARGUMENTS

-library *library_name*

Specifies the name of the Milkyway library where the hard macro model resides.

-cell *cell_name*

Specifies the name of the cell.

-lib_cell *lib_cell_list*

Specifies a list of hard macro library cells to be reported. By default, the command reports information for all modeled hard macro cells. This option is mutually exclusive with the -macro_name option.

-macro_name *macro_name_list*

Specifies a list of hard macros to be reported. By default, the command reports the information for all modeled hard macro cells. This option are mutually exclusive with the -lib_cell option.

-loaded_model_list

Reports a list of hard macros only. The option is used when the user wants to see the list of the loaded hard macro models.

-unmodeled_macro_list

Reports a list of macro cells which do not have rail macro models.

-include_unmodeled

Reports the unmodeled cells that are not loaded into the memory using the **open_mw_cel** command. Usually a hard macro is not loaded if it is not modeled or its macro model file is corrupted.

-report_instances

Reports cell instances that are referenced to the specified hard macros.

-pin_detail

Reports the details about the internal pins of the specified hard macros, such as pin weights and locations.

-internal_net_detail

Reports the details about the on-state resistances of the internal net.

-mode_detail

Reports the detailed condition mode information for the specified hard macros. By default, the command includes only a brief summary on the condition mode.

-dump_mode_waveform

Saves the mode-based current waveforms into a waveform file in the name specified by the -waveform_filename option. By default, the output file is saved in the FSDB format. To save the output file in the ASCII format, use the -waveform_filetype option. When specified, the output report does not contain the hard macro information.

-waveform_filename *file_name*

Specifies the name of the mode-based waveform file to be generated. The default is *hard_macro_mode_waveform.fsdb* or *hard_macro_mode_waveform.txt*, depending on the setting of the -waveform_filetype option. This option must be used with the -dump_mode_waveform option.

-waveform_filetype *fsdb* | *text*

Saves the mode-based waveform file in the FSDB or ASCII format. The default is *fsdb*. This option must be used with the -dump_mode_waveform option.

DESCRIPTION

This command reports the contents for the rail macro models that generated by the **create_rail_macro_model** command. The following is an example of the report:

Macro model data

- Macro cell name

- Number of ports
- Physical model
  - The only physical model type that is supported currently
- Electrical model
  - The current source distribution information that is attached to the CONN view. Currents are either extracted from timing and power analysis or assigned by the user.
  - Customized electrical model for current events
- Model process/voltage/temperature

Macro power/ground port data

- Port name
- The number of pins - the pins that represent current sources

Port electrical model data

- Port power type
  - Weighted only - The port electrical model electrical model.
  - Mode-based power - The electrical model based on current events.
- Port power distribution - The way that currents are distributed to pins
  - event
  - weighted

Port power mode data (available when the port power type is mode-based)

- Mode name
- Average current, max current, and duration

Detailed mode information (available when the -mode_detail option is defined)

- Mode name
- Trigger signal(s) - the trigger signal, such as clk
- Trigger edge - Rising/Falling
- Active condition - the WHEN expression of the event
- Average current
- Duration
- Max current
- Min current

Pin detailed data (available when the -pin_detail option is defined)

- Port total weight
- Individual pin weight and location. Pin weight is not reported if the current is evenly distributed to the

ports.)

Waveform file dump (available when the -dump_mode_waveform option is defined). The mode-based current waveform report contains the following information:

- Hard macro name
  - Hard macro electrical model port and PVT
    - Hard macro mode waveform

EXAMPLES

The following example reports the information for all hard macros in the design.

```
pr_shell> report_rail_macro_model

****************************************
Report : Modeled Hard Macros
Design : chip
Version: H-2012.12
Date   : Mon Nov 19 01:44:01 2012
****************************************

================================================================================
Hard Macro sram1056x12 Report
================================================================================

Hard Macro Model :

  Macro Name          Ports Physical Model  Electrical Model
  ---------------------------------------------------------------------------
  sram1056x12         3     CONN            DWM-lite
  ---------------------------------------------------------------------------

  Model Operating Condition :

  Process     Voltage(V)     Temperature(Celsius)
  ---------------------------------------------------------------------------
  1.000       1.500          27.000
  ---------------------------------------------------------------------------

Port VSS Content :

  Port Name      Number of Pins
  ---------------------------------------------------------------------------
  VSS            19670
  ---------------------------------------------------------------------------

  Electrical Model Data :

  Power Type     Distribution
  ---------------------------------------------------------------------------
  Mode-based power Weighted
  ---------------------------------------------------------------------------

  Mode Current Summary :

  Mode Name   Average(A)     Max(A)       Duration(s)
  ---------------------------------------------------------------------------
  Read        0.0019         0.0343       6.41e-09
```

```
  Write       0.00249        0.0367      6.41e-09
  ---------------------------------------------------------------------------

Port VDD Content :

  Port Name      Number of Pins
  ---------------------------------------------------------------------------
  VDD            11943
  ---------------------------------------------------------------------------

  Electrical Model Data :

  Power Type     Distribution
  ---------------------------------------------------------------------------
  Mode-based power Weighted
  ---------------------------------------------------------------------------

  Mode Current Summary :

  Mode Name   Average(A)     Max(A)      Duration(s)
  ---------------------------------------------------------------------------
  Read        -0.0019        -0.0343     6.41e-09
  Write       -0.00249       -0.0367     6.41e-09
  ---------------------------------------------------------------------------
```

The following example reports hard macro information in a list only.

```
pr_shell> report_rail_macro_model -loaded_model_list
****************************************
Report : Modeled Hard Macros
Design : chip
Version: H-2012.12
Date   : Mon Nov 19 01:45:35 2012
****************************************

================================================================================
Hard Macro List
================================================================================

sram1056x12
```

The following example reports the information for the hard macros, including the internal pin information.

```
pr_shell> report_rail_macro_model -pin_detail

****************************************
Report : Modeled Hard Macros
Design : TEST_DESIGN
Version: ...
Date   : ...
****************************************

================================================================================
Hard Macro sram1056x12 Report
================================================================================

Hard Macro Model :

  Macro Name          Ports
  ---------------------------------------------------------------------------
  sram1056x12         2
  ---------------------------------------------------------------------------

  Model Operating Condition :
```

```
  Process     Voltage(V)     Temperature(Celsius)
  ---------------------------------------------------------------------------
  1.000       1.500          27.000
  ---------------------------------------------------------------------------

Port VSS Content :

  Port Name      Number of Pins
  ---------------------------------------------------------------------------
  VSS            19670
  ---------------------------------------------------------------------------

  Electrical Model Data :

  Power Type     Distribution   Total Capacitance
  ---------------------------------------------------------------------------
  Mode-based power Weighted     2.6105e-11
  ---------------------------------------------------------------------------

  Pin Weight List :

  ID    Location             Weight         Pin Capacitance
  ---------------------------------------------------------------------------
  1     [13] (144780, 324890) 5.08388e-05   1.3271e-15
  2     [13] (140860, 324890) 5.08388e-05   1.3271e-15
  3     [13] (139820, 324890) 5.08388e-05   1.3271e-15
  4     [13] (145420, 324890) 5.08388e-05   1.3271e-15


------------------(The rest of the report is omitted.)--------------
```

The following example reports the information for the hard macros, including the internal pin information.

```
pr_shell> report_rail_macro_model -internal_net_detail

  Internal Net Data :

  Internal Net Name    Number of Current Source
  ---------------------------------------------------------------------------
  vddpi                13938
  ---------------------------------------------------------------------------

  Switch Cell List :
  Ext. Net Pin         Int. Net Pin         R-ON Resistance
  ---------------------------------------------------------------------------
  [31] (10905, 93615)  [31] (11040, 94585)  27.117
  [31] (1005, 21612)   [31] (1140, 21612)   22.597
  [31] (10467, 8015)   [31] (10002, 8265)   135.58
------------------(The rest of the report is omitted.)--------------
```

The following example writes the mode-based hard macro information to an output current waveform file in the ASCII format.

```
pr_shell> report_rail_macro_model -dump_mode_waveform -waveform_filetype text
Writing sram1056x12 waveform ...
All Macro waveforms successfully dumped to hard_macro_mode_waveform.txt
1

Here is an example of the hard_macro_mode_waveform.txt file:

* Hard macro mode waveform report file
* Time resolution: 1e-12
* Current unit: A
* Macro sram1056x12
```

```
*   Electrical model P = 1.000, V= 1.000, T = 27.000
*     Port VSS
* mode "Read" total pwl current waveform
( time, value )
( 0, 0.00013586 )
( 5e-12, 0.00017436 )
( 1e-11, 0.00022243 )
------------------(ignored PWL waveforms)--------------
* mode "Write" total pwl current waveform
The content of hard_macro_mode_waveform.txt:
* Hard macro mode waveform report file
* Time resolution: 1e-12
* Current unit: A
* Macro sram1056x12
*   Electrical model P = 1.000, V= 1.000, T = 27.000
*     Port VSS
* mode "Read" total pwl current waveform
( time, value )
( 0, 0.00013586 )
( 5e-12, 0.00017436 )
( 1e-11, 0.00022243 )
------------------(ignored PWL waveforms)--------------
* mode "Write" total pwl current waveform
( time, value )
( 0, 0.00014528 )
( 5e-12, 0.0001777 )
------------------(The rest of the report is omitted.)--------------
```

The following example reports the information for the hard macro that is saved in the cell name is sram1056x12 and the Milkyway library is memlib2.

```
pr_shell> report_rail_macro_model -library memlib2 -cell sram1056x12
```

SEE ALSO

open_mw_cel(2)
create_rail_macro_model(2)
rail_macro_model_flow(2)

report_rail_power

Reports the power value that has been set by the **set_rail_power** command.

SYNTAX

```
status report_rail_power
   object_list
```

Data Types

```
list object_list
```

ARGUMENTS

object_list

A list of cells, lib_cells or pg_pins on which to report the power set by the **set_rail_power** command.

DESCRIPTION

The command provides a report of the power that has been set by the **set_rail_power** command. The command returns 1 if successful, and 0 if unsuccessful.

The report includes the specific objects that have power assigned by **set_rail_power**, the calculated power values by the **update_currents** command, and the assigned power values. It also indicates whether the power for the PG pin is indirectly assigned or not using "derived". For example, if the user assigns power to a cell instance, then the power for the PG pin of this cell instance is considered as derived and indirectly assigned.

If the object_list is not specified, all the PG pins with the power set by **set_rail_power** will be reported.

EXAMPLES

The following example shows a sample report for all the PG pins with applied power, where the power for the vdd pin of cell instance U1/MyMult10/C5 is derived and indirectly assigned.

```
pr_shell> report_rail_power

Cells                 PG_Pin_Name    Calculated_Power    Assigned_Power
```

```
-------------------------------------------------------------------------------
U1/MyMult10/C5        vdd            .01                 0.5            derived
U1/U0                 vdd1           .03                 0.2
U1/U0                 vdd2           .03                 0.2
U1/U0                 vdd3           .03                 0.2
```

The following example shows a sample report for all PG pins named vdd2:

```
pr_shell> report_rail_power [get_pg_pins */vdd2]

Cells                 PG_Pin_Name    Calculated_Power    Assigned_Power
-------------------------------------------------------------------------------
U1/U0                 vdd2           .03                 0.2
U2/U0                 vdd2           .03                 0.25
```

SEE ALSO

rail_attributes(2)
remove_rail_power(2)
set_rail_power(2)

report_rail_results

Reports all the rail results in the session.

SYNTAX

```
string report_rail_results
   [result_name]
   [-verbose]
```

Data Types

```
string result_name
```

ARGUMENTS

result_name

Specifies the name of the rail result to be reported. If no *result_name* is specified, the command reports information for all the available rail results in the session.

-verbose

Expands the report to include additional information about each rail result, including the type of analyses and the supply nets on which the analyses are performed, and a snapshot of taps when the result is created.

DESCRIPTION

Whenever a rail analysis calculation command, such as **calculate_rail_voltage** or **calculate_minimum_path_resistances,** is successfully run, a named rail result is created for saving the data generated by that analysis. The **report_rail_results** command reports the rail results included in the current session. All rail results are reported if no *result_name* is specified. The report can be limited to a single result by providing an optional *result_name.*

The default report for a rail result provides the name of the result. Use the *-verbose* option to include the analysis type, for example, rail voltage or minimum path resistance, and the names of the supply nets in the report, and a snapshot of taps when the result is created.

EXAMPLES

The following example produces a report showing that two results, MY_NAMED_VDROP_RAIL_RESULT and RAIL_RESULT_1, are available:

```
pr_shell> report_rail_results

****************************************
Report : rail analysis results
Design : micro
Version: D-2010.06-alpha1
Date   : Fri Feb 19 16:07:00 2010
****************************************

   MY_NAMED_VDROP_RAIL_RESULT
   RAIL_RESULT_1
1
```

The following example shows a verbose report that includes the supply nets contained within the rail results, and a snapshot of taps when the result is created:

```
pr_shell> report_rail_results -verbose

****************************************
Report : rail analysis results
Design : micro
Version: D-2010.06-alpha1
Date   : Fri Feb 19 16:08:00 2010
****************************************

   MY_NAMED_VDROP_RAIL_RESULT has rail voltage result
      Supply nets in the rail result: VDD VSS

      Taps in the result: TAP_1 TAP_2

   RAIL_RESULT_1 has minimum path resistance result
      Supply nets in the resistivity result: VDD VSS

      Taps in the result: TAP_3 TAP_4 TAP_5
1
```

SEE ALSO

create_taps(2)
calculate_minimum_path_resistances(2)
calculate_rail_voltage(2)
current_rail_result(2)
get_taps(2)
remove_rail_result(2)
rename_rail_result(2)
report_taps(2)
tap_contexts(2)
pr_work_dir(3)

report_rail_voltage

Reports the rail analysis results calculated for the current rail result.

SYNTAX

```
string report_rail_voltage
   [supply_nets]
   [-limit N]
   [-effective_voltage_drop_or_rise]
   [-primetime file_name]
```

Data Types

```
list supply_nets
int N
string file_name
```

ARGUMENTS

supply_nets

Specifies a list of supply nets for which rail analysis results are reported. Items within the list can be either supply net collections or name patterns. By default, the command reports on all supply nets in the current rail result.

-limit *N*

Reports on the top *N* PG rail analysis result per supply net. By default, the limit is set to 5. The maximum of this limit is 100. The command writes all rail analysis result to the rail_result.out file, if the limit is set to 0.

-effective_voltage_drop_or_rise

Reports the effective voltage drop or rise for the specified supply nets. The number of intances the command reports for each supply net is 5 for default; if the option -limit *N* is applied, the number of instance changes to *N*.

-primetime *file_name*

Writes PG pin voltages into the specified file, in the format of the **set_voltage** commands.

DESCRIPTION

The **report_rail_voltage** command reports the PG pin voltage drops or the effective voltage from the current rail result. A rail result is generated by a successful execution of a rail calculation command, such as

calculate_rail_voltage, and contains the data created by that analysis. See the **current_rail_result** man page for more information on managing rail results.

By default, the report generated by the **report_rail_voltage** command includes PG names and voltage drop values for the top 5 voltage drops for each supply net in the **current_rail_result**. The voltage drop values are in the current library units. Use the supply_net option to report on specific nets only. Use the -limit option to change the number of PG pins to be reported for each net. The PG pins are reported in a descending order, from high to low.

When the option -effective_voltage_drop_or_rise is specified, the command reports effective voltage values in the current rail result.

Note that the -primetime option is mutually excluded with the -limit option.

When the **report_macro_internal_pin** variable is set true, the command reports macro model internal pin voltages.

EXAMPLES

The following example reports the voltage values for the supply nets VDD and VSS. It is a static rail analysis.

```
pr_shell> report_ rail_voltage -limit 5

Top 5 static voltage drop values for supply net VDD at instance ports:
1: 693.088e-3 (mV) at buffd_G12I2/VDD (buff7)
2: 676.534e-3 (mV) at bufba_G12I6/VDD (buffd)
3: 670.242e-3 (mV) at oai211d1_W2/VDD (oai2d)
4: 656.981e-3 (mV) at bufbd_G1B2I/VDD (bufd7)
5: 648.552e-3 (mV) at instruction_reg/VDD (dnb1)

Top 5 static voltage rise values for supply net VSS at instance ports:
1: 791.158e-3 (mV) at buffd_G12I2/VSS (buff7)
2: 791.025e-3 (mV) at ai22d1_I178/VSS (ai22d)
3: 791.025e-3 (mV) at n02d0_T2368/VSS (n02d1)
4: 784.504e-3 (mV) at ar221d1_I21/VSS (ar21d)
5: 780.852e-3 (mV) at oim211d1_W2/VSS (oim1d)
1
```

The following example reloads a previously generated rail result and reports the voltage variation for the supply net VDD. It is a static rail analysis.

```
pr_shell> current_rail_result MyResult5
TopDesign VDD
TopDesign VSS

pr_shell> report_rail_voltage VDD -limit 5

Top 5 static voltage drop values for supply net VDD at instance ports:
1: 693.088e-3 (mV) at buffda_G1B2I2/VDD (buffd7)
2: 676.534e-3 (mV) at bufbda_G1B2I6/VDD (buffda)
3: 670.242e-3 (mV) at oaim211d1_W21/VDD (oaimd1)
4: 656.981e-3 (mV) at bufbda_G1B2I1/VDD (bufbd7)
5: 648.552e-3 (mV) at instruction_reg/VDD (dnb1)
1
```

The following example reports the top 5 voltage values for the supply net VSS.

```
pr_shell> report_rail_voltage VSS
```

```
Top 5 static voltage rise values for supply net VSS at instance ports:
1: 791.158e-3 (mV) at buffda_G1BI2/VSS (buffd7)
2: 791.025e-3 (mV) at aoi22d1_I182/VSS (aoi2d1)
3: 791.025e-3 (mV) at nd02d0_T2368/VSS (nd02d1)
4: 784.504e-3 (mV) at aor221_I2531/VSS (aor2d1)
5: 780.852e-3 (mV) at oaimd1_W2524/VSS (oaimd1)
1
```

The following example writes the PG pin voltages in the format of **set_voltage** commands into the pin_voltage.tcl file.

```
pr_shell> report_rail_voltage -primetime pin_voltage.tcl
```

SEE ALSO

calculate_rail_voltage(2)
current_rail_result(2)
report_rail_results(2)
report_units(2)
report_macro_internal_pin(3)

report_reference

Reports the references in current instance or design.

SYNTAX

```
string report_reference
   [-nosplit]
```

ARGUMENTS

-nosplit

Does not split lines if the column overflows.

DESCRIPTION

Displays information about all references in the current instance or current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the **current_design**.

EXAMPLES

This is an example of the **report_reference** output.

```
pt_shell> report_reference

****************************************
Report : reference
Design : middle
****************************************


Attributes:
    b - black box (unknown)
    h - hierarchical
    n - noncombinational

Reference       Library       Unit Area  Count  Total Area  Attributes
--------------------------------------------------------------------------------
FD2                tech_lib         3.00     4          12.00      b
ND2                tech_lib         3.00     4          12.00      b
```

```
inter                                6.00     2          12.00      h
low                                  2.00     2           4.00      h
--------------------------------------------------------------------------------
Total 4 references                                       40.00
```

SEE ALSO

report_hierarchy(2)
report_cell(2)

report_scale_parasitics

Reports the parasitic resistance and capacitance scaling factors previously set by the **scale_parasitics** command.

SYNTAX

```
status report_scale_parasitics
   [net_list]
```

Data Types

```
net_list     list
```

ARGUMENTS

net_list

Limits the reporting of net-specific scaling settings to only the listed nets. By default, the command reports all net-specific scaling settings, in addition to the global scaling settings.

DESCRIPTION

The **report_scale_parasitics** command reports the parasitic resistance and capacitance scaling factors previously set by the **scale_parasitics** command. Both the global and net-specific scaling settings are reported.

EXAMPLES

The following example reports the parasitic scaling factors.

```
pt_shell> report_scale_parasitics
Global factors:
Cap factor      : 1.400000
Res factor      : 1.500000
Coupling factor : 1.350000
Net                    Cap factor   Res factor   Coupling factor
-----------------------------------------------------------------
net_sdram_DQ_in[0]     1.200000     1.500000     1.200000
net_sdram_DQ_in[1]     1.200000     1.500000     1.200000
net_sdram_DQ_in[2]     1.200000     1.500000     1.200000
```

```
net_sdram_DQ_in[3]     1.200000     1.500000     1.200000
1
```

SEE ALSO

```
read_parasitics(2)
reset_scale_parasitics(2)
scale_parasitics(2)
```

report_sense

Reports user-specified unatenes and sense propagation constraints for data or clock.

SYNTAX

```
string report_sense
   [-type clock | data]
   [-clocks clock_list]
   [-nosplit]
   object_list
```

Data Types

```
clock_list       list
object_list      list
```

ARGUMENTS

-type clock | data

Specifies whether the type of sense applies to clock or data networks. The allowed values are **clock** (the default) and **data**.

-clocks *clock_list*

Reports the specified clock objects. If you do not use the **-clocks** option, the tool reports all clocks passing through the given pin or arc objects.

-nosplit

Prevents line splitting if a column overflows.

object_list

Lists of ports, pins, or cell timing arcs to report.

DESCRIPTION

This command reports the user-defined constraints applied to the senses and unateness propagation of data and clock signals. These reports are distinguished based on whether the sense narrows down the clock waveform, or the sense creates a new clock waveform, as specified by the attributes G for source latency waveform and P for primary clock waveform.

EXAMPLES

The following example specifies and reports a stop sense propagation through an arc:

```
pt_shell> set_sense -stop_propagation [get_timing_arcs -from u2/A -to u2/Z] \
          -clock [get_clock clk12]
pt_shell> update_timing
pt_shell> report_sense -type clock

****************************************
Report : sense
         -type clock
Design : simple_2ff
 ...
****************************************


Attributes:
    G - Generated
    P - Primary
Object                                   Clock         Sense           Attr
--------------------------------------------------------------------------------
u2/A -> Z                                clk12         stop_prop
1
```

The following example selects positive unateness for pin MUX1/Z for all clocks:

```
pt_shell> set_sense -positive -clock clk MUX1/Z
pt_shell> update_timing
pt_shell> report_sense
****************************************
Report : sense
         -type clock
Design : nonunate
 ...
****************************************

Object                    Clock                Sense
-------------------------------------------------------------
MUX1/Z                       *                 positive
```

SEE ALSO

```
remove_sense(2)
set_sense(2)
```

report_si_aggressor_exclusion

Reports the exclusive groups set by the **set_si_aggressor_exclusion** command.

SYNTAX

```
status report_si_aggressor_exclusion
   [-rise]
   [-fall]
   [-nosplit]
   [net_list]
```

Data Types

```
net_list       list
```

ARGUMENTS

-rise

Reports all the exclusive groups that have been set as exclusive in the rise direction. If neither the **-rise** nor the **-fall** options are specified, the exclusive groups of the aggressor nets *anets* in both rise and fall directions are reported.

-fall

Reports all the exclusive groups that have been set as exclusive in the fall direction. If neither the **-rise** nor the **-fall** options are specified, the exclusive groups of the aggressor nets *anets* in both rise and fall directions are reported.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

net_list

Specifies the list of nets for exclusive groups you want reported. You can use this option with either the **-rise** or **-fall** options to specify exclusive groups only in that particular direction.

DESCRIPTION

The **report_si_aggressor_exclusion** command reports all the exclusive groups that have been set by the

set_si_aggressor_exclusion command.

EXAMPLES

The following example shows how the exclusion on nets *SCAN_LOGIC** for rise direction is reported.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
pt_shell> report_si_aggressor_exclusion
****************************************
  Report : report_si_aggressor_exclusion
           -rise
  Design : top
  Version: X-2005.12-DEV
  Date   : Thu Aug 11 14:43:46 2005
  ****************************************

  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ---------------------------------------------------------------------------
  SCAN_LOGIC1 SCAN_LOGIC2  R                  1
1
```

The following example shows how the exclusive groups on specific nets can be reported for rise or fall direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets {Agg1 Agg2 Agg3}]
-number_of_active_aggressors 2 -rise
1
pt_shell> set_si_aggressor_exclusion [get_nets {Agg2 Agg4}]
-number_of_active_aggressors 1 -fall
1
pt_shell> report_si_aggressor_exclusion [get_nets Agg2] -rise
****************************************
  Report : report_si_aggressor_exclusion
           -rise
           -fall
  Design : top
  ...
  ****************************************

  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ---------------------------------------------------------------------------
  Agg1 Agg2 Agg3          R                   2
1

pt_shell> report_si_aggressor_exclusion [get_nets Agg2] -fall
****************************************
  Report : report_si_aggressor_exclusion
           -rise
           -fall
  Design : top
  ...
  ****************************************
```

```
  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ---------------------------------------------------------------------------
  Agg2 Agg4               F                   1
1
```

SEE ALSO

```
remove_si_aggressor_exclusion(2)
report_delay_calculation(2)
report_noise_calculation(2)
set_si_aggressor_exclusion(2)
set_si_delay_analysis(2)
set_si_noise_analysis(2)
si_analysis_logical_correlation_mode(3)
```

report_si_bottleneck

Identifies the crosstalk bottlenecks in the design. This is useful when the major sources of violations come from crosstalk effects.

SYNTAX

```
status report_si_bottleneck
   [-cost_type type]
   [-slack_lesser_than slack]
   [-max_nets number]
   [-significant_digits digits]
   [-nosplit]
   [-include_clock_nets]
   [-minimum_active_aggressors active_aggressor_count]
   [-min]
   [-max]
   [-pre_commands pre_command_string]
   [-post_commands post_command_string]
```

Data Types

type	float
slack	float
number	int
digits	int
active_aggressor_count	int
pre_command_string	string
post_command_string	string

ARGUMENTS

-cost_type *type*

Sorts nets based on the specified cost types. The following three cost types return a list of victim nets:

- **delta_delay** - This cost factor is calculated by determining the worst delta delay on the victim net. This is useful for finding the largest delta delays in the design that contributes to failing paths.
- **delta_delay_ratio** - This cost factor is calculated by determining the worst delta delay ratio (ratio of delta delay to total stage delay) on the victim net. This cost factor is useful for finding the delta delays that result in the largest percentage increase of a stage delay in a failing path.
- **total_victim_delay_bump** - This cost factor is calculated by determining the height of the largest crosstalk bump on the victim net. This cost factor is useful for finding the strongest electrical couplings in the design that contributes to failing paths.

The following cost type returns a list of aggressor nets:

- **delay_bump_per_aggressor** - This cost factor is calculated for an aggressor net by finding all slack-qualifying victim nets that it affects, then summing up the bump voltages on those victims induced by the aggressor. This cost factor is useful for finding aggressors that result in the largest amount of electrical coupling to victim nets in failing paths.

-slack_lesser_than ***slack***

Applies the cost function only to victim nets with slack less than the specified limit.

-max_nets ***number***

Specifies the maximum number of nets to report. The default is 20.

-significant_digits ***digits***

Specifies the number of digits to display; the allowed range is 0 to 13. The default is specified by the **report_default_significant_digits** variable.

-nosplit

Does not split lines if column overflows.

-include_clock_nets

Includes the clock nets for bottleneck. By default, clock nets are excluded from the bottleneck analysis.

-minimum_active_aggressors ***active_aggressor_count***

Specifies the minimum number of active aggressors for the net to be selected. The default is 1.

-min

Specifies minimum (setup) analysis. For example, the slack limit applies to minimum slack of the net in the net selection.

-max

Specifies maximum (hold) analysis. For example, the slack limit applies to maximum slack of the net in the net selection.

-pre_commands ***pre_command_string***

Specifies a string of commands to be executed in the worker context before the execution of the merged reporting command. Commands must be grouped using the ";" character. The maximum size of a command is 1000 characters. This option is available only if you invoked PrimeTime with the **-multi_scenario** option.

-post_commands ***post_command_string***

This option allows you to specify a string of commands to be executed in the worker context after the execution of the merged reporting commands. Commands are grouped using the ";" character. The maximum size of a command is 1000 characters. This option is available only if you invoked PrimeTime with the **-multi_scenario** option.

DESCRIPTION

This command report the nets that are PrimeTime SI bottlenecks, either as a victim or aggressor. The cost functions are defined and reported on nets. The cost function is computed by considering only victims that have a slack lesser than the *slack* value. If neither the **-min** nor **-max** options are specified, both minimum and maximum

slacks are considered.

If a victim meets the slack criteria only for one of its transition senses (rise or fall) but not for the other, only the PrimeTime SI information for the failing sense is considered.

By default, the clock nets are not included in the bottleneck victim analysis, as the slack of the clock nets is infinite. However, clock nets can be included as victims in the bottleneck search by specifying the **-include_clock_nets** option. Even without this option, clock aggressor nets can be returned by the *total_victim_delay_bump* cost type.

Crosstalk problems can be caused by one or more of the following factors:

- Large coupling capacitance
- Weak victim net driver or slow victim transition
- Strong aggressor net driver

Depending on the relative victim and aggressor slacks, one of the following repairs could be considered:

- Downsize the aggressor net driver to a weaker driver
- Upsize the victim net driver to a stronger driver
- Model a physical decoupling using the **set_coupling_separation** command
- Insert a buffer on the victim net to break up long coupled routes

When sizing cells, it can be useful to use the **get_alternative_lib_cells** command to report equivalents, or the **report_alternative_lib_cells** command to evaluate their potential slack improvements.

The aggressor-based cost factor can be useful for finding strong aggressors which attack many failing victim nets. If the aggressor has positive setup slack, it might be preferable to downsize the aggressor driver and weaken its effect over multiple victims, or to separate it from the victims using the **set_coupling_separation** command. When performing such an aggressor downsizing, ensure that the resulting weaker aggressor driver does not violate any library max_capacitance or max_transition DRC requirements.

When a large number of aggressors attacks a victim net, you can take advantage of the statistical nature of many aggressors where the chances of all of them switching at the same time are very low. To find the failing victim nets that are attacked by many aggressors, the **-minimum_active_aggressors** option can be applied. The number of active aggressors is defined as the number of effective aggressors which are active for the worst-case aggressor alignment. For example, a victim net can have ten effective aggressors. However, for any given min/max rise/fall scenario, perhaps no more than five of these ten effective aggressors are active in the worst-case alignment for that sense. Such a victim would not meet a **-minimum_active_aggressors** value higher than five.

If you want to exclude specific crosstalk victim/aggressor interactions from the analysis, the **set_si_delay_analysis -exclude** command can be used to achieve this without changing the parasitics or the design.

To perform this analysis variable **timing_save_pin_arrival_and_slack** should be set to **true** when this command is issued. If this variable has not been set to **true**, before the command executes, it automatically sets it to **true** and updates the design timing. If you intend to use this command, it is recommended that you set this variable to **true** before the first timing update, therefore preventing the cost of an additional timing update.

EXAMPLES

The following examples show a PrimeTime SI bottleneck report. arc with specified startpoints and endpoints.

```
pt_shell> report_si_bottleneck -cost_type delta_delay -significant_digits 6

****************************************
Report : si_bottleneck
        -cost_type delta_delay
        -slack_lesser_than 0
        -max_nets 1
        -significant_digits 6
        -number_of_active_aggressors 1
        -min
        -max
Design : TestBH
****************************************

 Bottleneck Cost:  delta_delay
 net                                                          cost
 ---------------------------------------------------------------------
 InReg                                                        0.000007
```

SEE ALSO

set_coupling_separation(2)
size_cell(2)
report_default_significant_digits(3)

report_si_delay_analysis

Generates a report of user coupling information on nets for crosstalk delay analysis.

SYNTAX

```
int report_si_delay_analysis
   [-ignored_arrival]
   [-excluded]
   [-coupling_separated]
   [-disabled_statistical]
   [-nosplit]
   [nets]
```

Data Types

nets	list

ARGUMENTS

-ignored_arrival

Reports the list of nets you have specified to be analyzed with infinite window.

-excluded

Reports the list of nets excluded by you from crosstalk analysis as victim nets or aggressor nets. Also, specifies the analysis type for which the victim/aggressor nets are excluded. It indicates whether the information is set globally on a victim for all aggressors, or globally on a aggressor for all its victims, or between a particular victim and aggressor net. Indicates the type of analysis - minimum path victim falling, minimum path victim rising, maximum path victim falling and maximum path victim rising analysis. The combination of analysis types for which the net is excluded is reported.

-coupling_separated

Reports the list of nets on which you have specified coupling separation constraint. It reports the aggressor nets which are separated from all their victims, the victim nets that are separated from all their aggressors, and separation applied between a particular pair of victim and aggressor net.

-disabled_statistical

Reports the nets which you have disabled for statistical analysis when the nets are considered as composite aggressor.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field

exceeds the column width, the next field begins on a new line, starting in the correct column.

nets

Limits a list of nets in the current design for which the report needs to be generated.

DESCRIPTION

Produces a report showing coupling information specified by you on nets for crosstalk analysis.

The **report_si_delay_analysis** command allows you to view all of the sets of nets on which you have indicated the default crosstalk analysis behavior to be ignored and the behavior specified by you to take precedence. For every pair of nets, the type of behavior and analysis for which the default behavior is ignored, is reported. This command also specifies whether the information is specified for the net as a victim or aggressor.

Any combination of the options can be used to generate the corresponding report information. If no options are specified, the command reports all the information on all the nets.

This command reports the behavior set by the **set_si_delay_analysis**, **remove_si_delay_analysis**, **set_coupling_separation**, **remove_coupling_separation**, **set_si_delay_disable_statistical** and **remove_si_delay_disable_statistical** commands.

EXAMPLES

The following example shows how the excluded option can be used to report all the pairs of victims or aggressors that have been excluded from crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET*]
1
pt_shell> set_si_delay_analysis -exclude -agressors [get_nets AGG_NET*]
1
pt_shell> set_si_delay_analysis -exclude -victims [get_nets V_NET]
-aggressors [get_nets A_NET]
1

pt_shell> report_si_delay_analysis -excluded

****************************************
Report : report_si_delay_analysis
        -excluded
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis
```

```
Victim net              Aggressor net            Delay Analysis attributes
--------------------------------------------------------------------------------
CLK_NET1                *                        E{ mr mf Mr Mf }
CLK_NET2                *                        E{ mr mf Mr Mf }
CLK_NET3                *                        E{ mr mf Mr Mf }
*                       AGG_NET1                 E{ mr mf Mr Mf }
*                       AGG_NET2                 E{ mr mf Mr Mf }
V_NET                   A_NET                    E{ mr mf Mr Mf }
1
```

The '*' indicates all victim nets/ aggressor nets respectively.

The following example shows how to report all nets in a design which you have specified to be analyzed as infinite window.

```
pt_shell> set_si_delay_analysis -ignore_arrival [get_nets LOGIC0]
1
pt_shell> report_si_delay_analysis -ignored_arrival

****************************************
Report : report_si_delay_analysis
        -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis attributes
--------------------------------------------------------------------------------
LOGIC0                  *                        I
*                       LOGIC0                   I
1
```

The following example shows how to report all nets in a design on which you have specified a coupling separation constraint.

```
pt_shell> set_coupling_separation [get_nets LOGIC1]
1
pt_shell> report_si_delay_analysis -coupling_separated

****************************************
Report : report_si_delay_analysis
        -coupling_separated
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
```

```
     Mf - Maximum fall analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis attributes
--------------------------------------------------------------------------------
LOGIC1                  *                        P
*                       LOGIC1                   P
1
```

The following example shows how to report all nets in a design which you have specified to be disabled from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_delay_disable_statistical [get_nets LOGIC1]
1
pt_shell> report_si_delay_analysis -disabled_statistical

****************************************
Report : report_si_delay_analysis
        -disabled_statistical
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis attributes
--------------------------------------------------------------------------------
LOGIC1                  *                        T
*                       LOGIC1                   T
1
```

The following example shows how to limit the report to a list of nets in a design on which you have specified any coupling information for crosstalk delay analysis.

```
pt_shell> report_si_delay_analysis [get_nets V1]

****************************************
Report : report_si_delay_analysis
        -coupling_separated
        -disabled_statistical
        -excluded
        -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
```

```
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis attributes
--------------------------------------------------------------------------------
V1                      *                        I
*                       V1                       I T
V1                      V2                       E{ mr mf Mr Mf }
1
```

SEE ALSO

```
set_si_delay_analysis(2)
remove_si_delay_analysis(2)
set_coupling_separation(2)
remove_coupling_separation(2)
set_si_delay_disable_statistical(2)
remove_si_delay_disable_statistical(2)
```

report_si_double_switching

Reports the double switching violation detected in the design.

SYNTAX

```
status report_si_double_switching
   [-clock_network]
   [-rise]
   [-fall]
   [-nosplit]
   [nets]
```

Data Types

```
nets     list
```

ARGUMENTS

-clock_network

Reports the double switch violations for the clock network only, even if the **si_xtalk_double_switching_mode** variable is set to **clock_network** or **full_design**.

-rise

Reports the double switch violations for the rising victim.

-fall

Reports the double switch violations for the falling victim. If you use neither the **-rise** nor **-fall** option, the net is reported only one time with the smallest slack.

-nosplit

Suppresses line splitting if the line exceeds 80 characters.

nets

Reports for the specific victim nets. By default, all nets are reported.

DESCRIPTION

This command reports the victim nets with double switching violations. It reports the victim net name, actual bump height, the required bump height, and the double switching slack.

To use this command, you must enable double switching detection by setting the **si_xtalk_double_switching_mode** variable before the **update_timing** command.

EXAMPLES

The following example reports all double switching violations in the design for both rise and fall.

```
pt_shell> report_si_double_switching -nosplit -rise -fall

****************************************
Report : si_double_switching
        -nosplit
        -rise
        -fall
 ...
****************************************

  Victim   Switching   Actual        Required      Double Switching
  Net      Direction   Bump Height   Bump Height   Slack
  ------   ---------   -----------   -----------   ----------------
  I2       max_rise    0.69          0.42          -0.26 (VIOLATING)
  I2       max_fall    0.51          0.49          -0.02 (VIOLATING)
```

SEE ALSO

si_enable_analysis(3)
si_xtalk_double_switching_mode(3)

report_si_noise_analysis

Generates a report of user coupling information on nets for crosstalk noise analysis.

SYNTAX

```
int report_si_noise_analysis
   [-ignored_arrival]
   [-excluded]
   [-coupling_separated]
   [-disabled_statistical]
   [-nosplit]
   [nets]
```

Data Types

nets	list

ARGUMENTS

-ignored_arrival

Reports the list of nets you have specified to be analyzed with infinite window for noise analysis.

-excluded

Reports the list of nets excluded by you from noise analysis as victim nets or aggressor nets. Also specifies the analysis type for which the victim/aggressor nets are excluded. The analysis type may be above low rail, below low rail, above high rail or below high rail. The combination of analysis types for which the net is excluded is reported. It indicates whether the information is set globally on a victim for all aggressors, or globally on a aggressor for all its victims, or between a particular victim and aggressor net.

-coupling_separated

Reports the list of nets on which you have specified coupling separation constraint. It reports the aggressor nets which are separated from all their victims, the victim nets which have been separated from all their aggressors, and coupling separation applied between a particular victim and aggressor net.

-disabled_statistical

Reports the nets which you have disabled for statistical analysis when the nets are considered as composite aggressor.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

nets

Limits a list of nets in the current design for which the report needs to be generated.

DESCRIPTION

Produces a report showing coupling information specified by you on nets for noise analysis.

The **report_si_noise_analysis** command allows you to view all the set of nets on which you have indicated the default noise analysis behavior to be ignored and the behavior specified by you to take precedence. For every pair of nets, the type of behavior and analysis for which the default behavior is ignored, is reported. This command also specifies whether the information is specified for the net as a victim or aggressor.

Any combination of the options can be used to generate the corresponding report information. If no options are specified, the command reports all the information on all the nets.

This command reports the behavior set by the **set_si_noise_analysis**, **remove_si_noise_analysis**, **set_coupling_separation**, **remove_coupling_separation**, **set_si_noise_disable_statistical** and **remove_si_noise_disable_statistical** commands.

EXAMPLES

The following example shows how the excluded option can be used to report all the pairs of victims or aggressors that have been excluded from noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET*]
1
pt_shell> set_si_noise_analysis -exclude -agressors [get_nets AGG_NET*]
1
pt_shell> set_si_noise_analysis -exclude -victims [get_nets V_NET]
-aggressors [get_nets A_NET]
1

pt_shell> report_si_noise_analysis -excluded

****************************************
Report : report_si_noise_analysis
        -excluded
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis
```

```
Victim net              Aggressor net            Noise Analysis attributes
--------------------------------------------------------------------------------
CLK_NET1                *                        E{ al ah bl bh }
CLK_NET2                *                        E{ al ah bl bh }
CLK_NET3                *                        E{ al ah bl bh }
*                       AGG_NET1                 E{ al ah bl bh }
*                       AGG_NET2                 E{ al ah bl bh }
V_NET                   A_NET                    E{ al ah bl bh }
1
```

The '*' indicates all victim nets/ aggressor nets respectively.

The following example shows how to report all nets in a design which you have specified to be analyzed as infinite window.

```
pt_shell> set_si_noise_analysis -ignore_arrival [get_nets LOGIC0]
1
pt_shell> report_si_noise_analysis -ignored_arrival

****************************************
Report : report_si_noise_analysis
        -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis attributes
--------------------------------------------------------------------------------
LOGICO                  *                        I
*                       LOGIC0                   I
1
```

The following example shows how to report all nets in a design on which you have specified a coupling separation constraint.

```
pt_shell> set_coupling_separation [get_nets LOGIC1]
1
pt_shell> report_si_noise_analysis -coupling_separated

****************************************
Report : report_si_noise_analysis
         -coupling_separated
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
```

```
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis attributes
--------------------------------------------------------------------------------
LOGIC1                  *                        P
*                       LOGIC1                   P
1
```

The following example shows how to report all nets in a design which you have specified to be disabled from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_noise_disable_statistical [get_nets LOGIC1]
1
pt_shell> report_si_noise_analysis -disabled_statistical

****************************************
Report : report_si_noise_analysis
        -disabled_statistical
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis attributes
--------------------------------------------------------------------------------
LOGIC1                  *                        T
*                       LOGIC1                   T
1
```

The following example shows how to limit the report to a list of nets in a design on which you have specified any coupling information for noise noise analysis.

```
pt_shell> report_si_noise_analysis [get_nets V1]

****************************************
Report : report_si_noise_analysis
         -coupling_separated
         -disabled_statistical
         -excluded
         -ignored_arrival
         -reselected
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
```

```
      bl - below low analysis
      bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis attributes
--------------------------------------------------------------------------------
V1                      *                        I
*                       V1                       I T
V1                      V2                       E{ al ah bl bh }
1
```

SEE ALSO

```
set_si_noise_analysis(2)
remove_si_noise_analysis(2)
set_coupling_separation(2)
remove_coupling_separation(2)
set_si_noise_disable_statistical(2)
remove_si_noise_disable_statistical(2)
```

report_spice_options

Reports set up and configuration options set by the **set_spice_options** command. These options are used in the transistor-level simulation for the cells that do not have characterized libraries.

SYNTAX

report_spice_options

ARGUMENTS

None

DESCRIPTION

Reports the set up and configuration options that are set by the **set_spice_options** command. These options are needed in the transistor-level simulation for the cells that do not have characterized libraries.

EXAMPLES

The following example first sets configuration options for the SPICE simulation process and then reports the option settings. .

```
pr_shell> set_spice_options \
            -top_spice_ckt_file "VR1.net" \
            -include_path "/dir/for/include/files"
            1

pr_shell> report_spice_options
Spice Exec = /full/path/spice/exec_name
Spice TopCktFile = VR1.net
Setup File = setup_file_name
Spice IncPath = /dir/for/include/files
Spice HdlPath = NULL
Spice deck file name for voltage regulator and core co-simulation = Core_VoltageRegulator_cosim.sp
There is no master name mapping and pin name mapping specified.
1
```

SEE ALSO

`set_spice_options(2)`

report_supply_net_hierarchy

Reports supply net hierarchy for the specified supply nets.

SYNTAX

```
report_supply_net_hierarchy
   supply_nets
   [-hierarchy_level level]
```

Data Types

supply net collection *supply_nets*

ARGUMENTS

supply_nets

Lists the supply nets to be reported. Each element of the list can be a supply net collection or a name pattern.

-hierarchy_level level

Indicates that the report is to show standard cell instance information, which includes reporting cells that connect the related supply nets.

DESCRIPTION

This command reports supply net hierarchy for the spcified supply nets. The design must be loaded into the session using the **open_mw_cel** command before **report_supply_net_hierarchy** is executed.

The supply net hierarchy is reported for each of the specified supply nets.

EXAMPLES

The following example shows how to report supply net hierarchy for the supply nets VDD and VSS.

```
pr_shell> report_supply_net_hierarchy [list VDD VSS]
*************************************
Report : Supply Net Hierarchy
Design : test_chip
```

```
Version: J-2014.06
Date   : Thr Mar 13 13:44:05 2014
****************************************

0 -  (Master design) (6): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
   1 - SUB2 (Master sub_3) (18): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
   1 - SUB1 (Master sub_2) (12): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
0 -  (Master design) (6): Supply ports: VSS Supply Nets: VSS Rail visible supply net: VSS
   1 - SUB2 (Master sub_3) (18): Supply ports: VSS Supply Nets: VSS Rail visible supply net: VSS
   1 - SUB1 (Master sub_2) (12): Supply ports: VSS Supply Nets: VSS Rail visible supply net: VSS

pr_shell> report_supply_net_hierarchy VDD -hierarchy_level 1
****************************************
Report : Supply Net Hierarchy
Design : test_chip
Version: J-2014.06
Date   : Thr Mar 13 14:00:05 2014
****************************************

0 -  (Master design) (6): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
   1 - SUB2 (Master sub_3) (18): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
   1 -   INV_sub1 (Master aw1in01d4.FRAM): PG pin: vbp vdd
   1 -   INV_sub3 (Master aw1in01d4.FRAM): PG pin: vbp vdd
   1 -   INV_sub5 (Master aw1in01d4.FRAM): PG pin: vbp vdd
   1 -   FEED0_1 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_2 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_3 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_4 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_5 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_6 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 - SUB1 (Master sub_2) (12): Supply ports: VDD Supply Nets: VDD Rail visible supply net: VDD
   1 -   INV_sub1 (Master aw1in01d4.FRAM): PG pin: vbp vdd
   1 -   INV_sub3 (Master aw1in01d4.FRAM): PG pin: vbp vdd
   1 -   ARCR0_1 (Master aw1sw2m5cpb02.FRAM): PG pin: vbp vdd
   1 -   FEED0_3 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_2 (Master aw1feedth8.FRAM): PG pin: vbp vdd
   1 -   FEED0_1 (Master aw1feedth8.FRAM): PG pin: vbp vdd
```

SEE ALSO

open_mw_cel(2)

report_supply_network

Provides a report on the connectivity between supply nets and their corresponding MTCMOS switch cells, as well as the relationship between supply nets through MTCMOS switch cells.

SYNTAX

```
string report_supply_network
   [supply_nets]
   [-verbose]
```

Data Types

```
list supply_nets
```

ARGUMENTS

supply_nets

Specifies a list of supply nets to be reported. Items within the list can be name patterns or collections of supply nets. The default is to report on all of the supply networks.

-verbose

Indicates that the report is to show verbose supply network information, which includes reporting on the MTCMOS switch cells that connect the related supply nets.

DESCRIPTION

The command provides a report on the relationship between the specified supply nets and the related supply nets. A supply net is related to another supply net when there are one or more MTCMOS switch cells directly connected to both nets. For instance, in a multi-power domain design, a supply net VDD may be related and provides current paths to several other supply nets, such as VDD1 or U1/VDD, through MTCMOS switch cells that control when the other supply nets are on or off. This command provides information about which supply nets relate to which other supply nets, and optionally on how MTCMOS switch cells indirectly connect them.

The default information reported for a supply net is the list of names of the supply nets that are directly related to the given supply net via MTCMOS switch cell objects. The verbose report adds the names of the MTCMOS switch cells connecting the nets, and the current on/off status of the MTCMOS switch cells. See the **set_annotated_mtcmos_switch_cell** man page about how to specify the on/off status.

If no *supply_nets* parameter is provided, the report includes all supply nets.

EXAMPLES

The following example reports the supply network relationships for supply net VDD:

```
pr_shell> report_supply_network VDD
****************************************
Report : report_supply_network
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************

--------------------------------------------------------------------------------
Supply Net : VDD
--------------------------------------------------------------------------------
Connected Supply Net : VDD1
Connected Supply Net : VDD2
```

The following example provides a verbose report for the supply network relationships for supply net VDD, including the MTCMOS switch cells and their current on/off status between supply nets VDD and VDD1:

```
pr_shell> report_supply_network -verbose VDD
****************************************
Report : report_supply_network
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************

--------------------------------------------------------------------------------
Supply Net : VDD
--------------------------------------------------------------------------------
Connected Supply Net : VDD1
  MTCMOS Switch Cell : switch_class_15 (state:on) ("sw1 & !sw2" state: on) ("!sw1 &sw2" state: off)
  MTCMOS Switch Cell : switch_class_20 (state:on) ("sw" state: on)
  MTCMOS Switch Cell : switch_class_21 (state:off) ("sw" state: off)

Connected Supply Net : VDD2
  MTCMOS Switch Cell : switch_class_15 (state:on) ("sw" state: on)
  MTCMOS Switch Cell : switch_class_20 (state:on) ("sw" state: on)
  MTCMOS Switch Cell : switch_class_21 (state:on) ("sw1 & !sw2" state: off) ("!sw1 & sw2" state: on)
```

The following example reports the supply network for all supply nets:

```
pr_shell> report_supply_network
****************************************
Report : report_supply_network
Design : sample_design
Version: D-2010.06
Date   : Fri Feb  5 17:17:46 2010
****************************************

--------------------------------------------------------------------------------
Supply Net : VDD
--------------------------------------------------------------------------------
No supply net connected to VDD through MTCMOS switch cell

--------------------------------------------------------------------------------
```

```
Supply Net : VSSAVG
--------------------------------------------------------------------------------
Connected Supply Net : VSSSWITCH

--------------------------------------------------------------------------------
Supply Net : VSSSWITCH
--------------------------------------------------------------------------------
Connected Supply Net : VSSAVG
Connected Supply Net : VSS

--------------------------------------------------------------------------------
Supply Net : VSS
--------------------------------------------------------------------------------
Connected Supply Net : VSSSWITCH
```

SEE ALSO

```
get_supply_nets(2)
report_supply_net(2)
set_annotated_mtcmos_switch_cell(2)
```

report_supply_parasitics

Reports the supply net parasitics on the current design.

SYNTAX

```
status report_supply_parasitics
   [supply_net_list]
   [-noexpand]
```

Data Types

```
list supply_net_list
```

ARGUMENTS

supply_net_list

Specifies the list of supply nets for which parasitics are reported. Each element on the list can be a supply net collection or name pattern. If no supply nets are specified, then all supply nets are included in the report.

-noexpand

Indicates that the command does not expand the *supply_net_list* to include the related supply nets that are indirectly connected to the listed supply nets through MTCMOS switch cells (also referred to as power management cells). By default, the list of supply nets is expanded.

DESCRIPTION

The command reports parasitics of the specified supply nets. The contents of the report include:

- Nber of resistive elements
- Number of inductive elements
- Number of capacitive elements
- Number of cell instance connections
- Total net capacitance in pF

If no supply nets are specified, all supply nets are included in the report. When specific nets are provided, the command automatically expands the list of supply nets to include the supply nets that are related to the listed supply nets through power switch (also known as MTCMOS or PM cells) connections. You can suppress this auto-

expansion by using the *-noexpand* option.

EXAMPLES

The following example reports parasitics of the supply nets VDD and VSS and their expanded related nets:

```
pr_shell> report_supply_parasitics {VDD VSS}

****************************************
Report : Supply parasitics
Design : ckt1
Version: D-2010.06
Date   : Mon May 24 14:13:19 2010
****************************************


Supply net "VDD" consists of:
        30 capacitive elements
        29 resistive elements
        26 cell instance connections
        total net capacitance 0.0505266 pF

Supply net "VSS" consists of:
        29 capacitive elements
        28 resistive elements
        26 cell instance connections
        total net capacitance 0.0409272 pF
```

SEE ALSO

```
read_supply_parasitics(2)
extract_supply_parasitics(2)
```

report_supply_set

Reports all supply sets defined in the design.

SYNTAX

```
status report_supply_set
```

ARGUMENTS

None.

DESCRIPTION

The **report_supply_set** command reports detailed information about the supply sets defined in the design. The report for a supply set includes the following information: its full name, the scope in which it was created, and the pairs of function and associated supply net.

This command returns 1 on success, otherwise a 0 returns.

EXAMPLES

The following example reports information about the primary_sset supply set created in the current scope.

```
prompt> create_power_domain top_domain
top_domain

prompt> create_supply_set sset \
          -function {power VDD} \
          -function {ground VSS}
sset

prompt> report_supply_set

  Total of 2 supply sets defined for design 'top'.

--------------------------------------------------------------------------------
    Supply Set : top_domain.primary
    Scope      : <top level>
    Function   : power, supply net association : top_domain.primary.power
```

```
    Function   : ground, supply net association : top_domain.primary.ground
    Associated : -
--------------------------------------------------------------------------------
    Supply Set : sset
    Scope      : <top level>
    Function   : power, supply net association : VDD
    Function   : ground, supply net association : VSS
1
```

SEE ALSO

create_supply_set(2)

report_supply_voltage

Reports voltage on the supply net used in rail analysis.

SYNTAX

```
status report_supply_voltage
   [list_of_supply_nets]
```

Data Types

```
list list_of_supply_nets
```

ARGUMENTS

list_of_supply_nets

A list of the supply nets on which to report voltage used in rail analysis.

DESCRIPTION

The **report_supply_voltage** command reports ideal voltages of the supply nets used in rail analysis, after rail analysis is done. The command returns 1 if successful and 0 if unsuccessful.

For each supply net in *list_of_supply_net*, this command outputs the supply net name, supply voltage, its source (assigned by the user or derived from the voltage of cell instance PG pins), and the number of cell instances connected to this supply net.

If the user performs **report_supply_voltage** before rail analysis, this command will report these supply nets whose supply voltages are assigned by command **set_supply_voltage**, since we derive the supply voltages from the voltages of cell instance PG pins only during rail analysis.

If *list_of_supply_net* is not given for the command, we will report the voltages for all the suppply nets in the design.

EXAMPLES

The following example reports the voltage for the supply net VDD used in rail analysis:

```
pr_shell> report_supply_voltage VDD
```

The following example reports the voltages for all the supply nets in the design:

```
pr_shell> report_supply_voltage
```

SEE ALSO

```
set_supply_voltage(2)
get_supply_nets(2)
```

report_switched_net_state

Reports the current state of the specified supply nets.

SYNTAX

```
status report_switched_net_state
   supply_list
```

Data Types

```
list supply_list
```

ARGUMENTS

supply_list

Specifies a list of supply nets for which to report the current state. For main supply nets in the specified list, the current states of their virtual supply nets are reported. For virtual supply nets in the specified list, their current states are reported.

DESCRIPTION

The command reports the current state of virtual supply nets in the design.

EXAMPLES

The following example sets the current state, *start_voltage* and *stop_voltage* for the virtual supply net VDDV. It reports the current state of virtual supply net VDDV.

```
pr_shell> set_switched_net_state -state ramp_up \
          -start_voltage 0.1 -stop_voltage 1.0 VDDV
1

pr_shell> report_switched_net_state VDDV
virtual supply net: VDDV
+ VDDV : state(on) start_voltage(  0.100) stop_voltage(  1.000) threshold (  0.010)
1
```

The following example reports the current state of the virtual supply nets of VDD.

```
pr_shell> report_switched_net_state VDD
main supply net: VDD
+ VDDV : state(ramp_up) start_voltage(  0.100) stop_voltage(  1.000) threshold (  0.010)
1
```

SEE ALSO

set_switched_net_state(2)

report_switching_activity

Reports the statistics on the switching activity annotation and signal probability on the current design or instance.

SYNTAX

```
integer report_switching_activity
   [-cells cells_list]
   [-list_not_annotated]
   [-list_annotated]
   [-list_low_activity]
   [-list_by_source source]
   [-base_clock clk]
   [-average_activity]
   [-coverage]
   [-hierarchy]
   [-show_pin]
   [-sort_by hier | net_toggle_rate | name]
   [-toggle_limit limit]
   [-exclude exclusion_group]
   [-include_only inclusion_group]
   [-only_related_clock clock]
```

Data Types

cells_list	list
source	string
clk	string
limit	integer
exclusion_group	string
inclusion_group	string
clock	clock

ARGUMENTS

-cells *cells_list*

Indicates that switching activity annotation is to be reported only for the specified cells_list.

-list_not_annotated

When the command is used with this flag, the report lists the design nets that have no user switching activity annotation. This report does not include constant value nets (logic one/zero nets). Note that such nets are annotated with default switching activity if they are not user annotated.

-list_annotated

When the command is used with this flag, the report lists the design nets that do have user switching activity annotation.

-list_low_activity

Reports a list of nets in the design with few toggles, where few is defined using the *-toggle_limit* option. The list report can be used to help debug which nets are not being exercised by the simulation test bench.

-list_by_source *source*

Reports a list of nets that have switching activity information from a given source.

Possible sources are any of the following:

[annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity]

Here *annotated* means any annotated (not propagated) source. The *file* source means any source with from an activity file (vcd or SAIF file).

-base_clock *clk*

When the *-average_activity* report is requested, average activities over nets are reported. The averaged toggle rates reported are by default with respect to 1ns. When the *-base_clock* option is used, the reported averaged toggle rates are with respect to the period of the clock *clk*. If the reported toggle rate is 0.5, and the period of the base clock is 3ns, then on average during simulation, there were 0.5 toggles every 3ns, or 1 toggle every 6 ns.

-average_activity

Produces a report that averages toggle rates over the nets in the design. When combined with the *-hierarchy* flag, average switching activity is computed for each subblock in the design.

It is possible using the filter options *-exclude* or *-include_only* to get a report of average toggle rates over a subset of the nets in the design. This can be useful, for instance, to get the average toggle rate for annotated nets, or even the average toggle rate for annotated nets driven by sequential cells.

-coverage

Causes the command to produce a summary report about the nets and the design that have switching activity information, but have few toggles. Coverage is defined as the percentage of nets with more than *limit* toggles.

See the *toggle_limit* option. You can use this report to verify that switching activities from simulation or propagation are properly exercising the design. Combined with the *-hierarchy* flag, you can use the report to make sure that each block in the design is properly exercised.

-hierarchy

When the command is used with this flag, the report generated includes information about subblocks in the design. The flag cannot be used with the list generating options for the command.

-show_pin

Specifies that driver pins as well as nets are shown in certain net lists. The lists generated by -list_not_annotated, -list_annotated, -list_low_activity, and -list_by_source lists are affected. Pins are not shown for multidriver nets.

-sort_by *hier* | *net_toggle_rate* | *name*

When used with the any of the hierarchical reports (that is, using the *-hierarchy* flag with the default report or with *-average_activity* or *-coverage* options), the hierarchical blocks in the report are sorted either by hierarchy (depth first), by averaged toggle rate, or by name.

When used with any of the list reports (that is, using *-list_low_activity* or *-list_by_source* or *-list_not_annotated* or

-list_annotated options), the list is sorted by toggle rate or by name. For the list reports, the *-sort_by hier* option is not accepted.

-toggle_limit *limit*

Sets the lower limit for what is considered to be few toggles. This limit is used by the *-coverage* report and by the *-list_low_activity* list report. The default limit is 1 toggle.

The value of the argument to the *-toggle_limit* option represents the maximum number of toggles considered to be low activity. The number of toggles on a net is defined to be the toggle rate times the simulation duration. For vector free power analysis (where no SAIF file or VCD has been used) the default simulation duration is 10000 ns. If a SAIF has been used, the duration is taken from the SAIF file. If a VCD file has been used to annotate activity, the duration is the duration of the VCD simulation.

-exclude *exclusion_group*

This filter option filters out nets from consideration before generating any of the reports the command can generate. Filtering occurs before toggle rate averaging, coverage percentage computations, and list generation. Possible exclusion groups are any of the following: *[sequential, combinational, primary_inputs, black_boxes, three_state, memory, clock_gate, clock, low_activity, high_activity, rtl]*.

A net is defined as sequential if it is driven by a sequential cell. Likewise with combinational, primary_inputs, etc.

The rtl exclusion group refers to nets driven either by sequential cell outputs, black box outputs, primary inputs, or memories. The rtl group is mainly provided as a shorthand for writing out all of these.

The clock_gate exclusion group refers to ICGs only, not other types of clock gates.

Other possible exclusion groups are defined by the source of the activity information:

[annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity]

Here *annotated* means any annotated (not propagated) source. The *file* source means any source from an activity file (usually vcd or SAIF file).

To exclude multiple groups, the argument *exclusion_group* can be a list with multiple groups in the list.

Note: Currently, the "sequential" group should not include nets driven by clock gates. In the C-2006.12 release, the group included such nets.

-include_only *inclusion_group*

This filter option filters out nets from consideration before generating any of the reports the command can generate. Filtering occurs before toggle rate averaging, coverage percentage computations, and list generation. Only nets that are members of the inclusion group are considered in the report generation.

Possible inclusion groups are any of the following: *[sequential, combinational, primary_inputs, black_boxes, three_state, memory, clock_gate, clock, low_activity, high_activity, rtl]*.

A net is defined as sequential if it is driven by a sequential cell. Likewise, with combinational, primary_inputs, etc.

The clock_gate inclusion group refers to ICGs only, not other types of clock gates.

Other possible inclusion groups are defined by the source of the activity information: *[annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity]*

Here *annotated* means any annotated (not propagated) source. The *file* source means any source from an activity file (usually VCD or SAIF file).

To include multiple groups, the argument *inclusion_group* can be a list with multiple groups in the list. In this case, only nets that are in one of the listed groups are included in the report.

To include only nets that are in several groups, use the & operator. This operator has low precedence; lists are or'ed first, then and'ed. For instance, to include only nets that are driven by RTL points or three_state buffers and have no switching activity, use the following:

pt_shell> **report_switching_activity -include_only {rtl,three_state & no_switching_activity}**

To invert the selection, the *!* operator can be used. For example, one way to include only nets that are driven by rtl points, and have switching activity, would be the following:

pt_shell> **report_switching_activity -include_only {rtl & !no_switching_activity}**

There may be other ways (possibly using the *-exclude* option) to get this same set of nets included in the report.

-only_related_clock *clock*

If specified, nets that have a base_clock other than *clock* are filtered out before the results of the report are included. This option can affect the overview, coverage, or the average toggle report. Also, this option can affect the output lists. Such reports consider only the nets with the given base_clock.

DESCRIPTION

Reports the switching activities on the nets in the current design or instance.

The default report generated by the **report switching_activity** command gives an overview of switching activity information in the current instance; the report lists the percentage of nets with switching activity and signal probability that is user-annotated, default-annotated, and propagated (if any).

Switching activity and signal probability can be annotated by using the **set_switching_activity** and **read_saif** commands, or by using event-simulation data using the **read_vcd** command. The **create_clock** and **set_case_analysis** commands also affect switching activities.

Note that the **read_vcd** command, when used with a named pipe or the *-pipe* option, does not immediately annotate activities on the design. The annotations happen later during the **update_power** command. Because of this, the **report_switching_activity** command does not report activity change in this situation until the **update_power** command has been run.

During power calculations, PrimeTime PX estimates the switching activity of objects that are not user-annotated by propagating the user or default annotated switching activity values. These objects are reported as propagated after the **update_power** command has been run, but as not propagated prior to running the **update_power** command. In general, propagated switching activity is less accurate than switching activity on objects derived by simulation, so the percentage values in the user-annotated column of the report general by the **report_switching_activity** command are an indication of the accuracy of power reports.

If the **power_analysis_mode** command has been set to *averaged* or *leakage_variation* and if a VCD file has been read with the **read_vcd** command without the *-pipe* option, the tool annotates the activity from the VCD prior to reporting the switching activity. In addition, unannotated objects where the activity can be derived accurately without random vector simulation (for example, clock nets, Qbar pins where the Q pin is annotated, constant nets) is reported by the **report_switching_activity** command as having *implied\P activity. Activity propagation for other unannotated nodes does not occur until the **update_power** command is run.*

If the **power_analysis_mode** has been set to *time_based*, the **get_switching_activity** command does not have access to toggle rate information prior to running the **update_power** command. The command reports whether a net is uninitialized or whether it is annotated from the VCD file, but since the VCD file has not been read

yet, the toggle rate and other values are reported as -1.

In some instances, you may have incomplete annotation from simulation. In this case, one or more nets may be unannotated after reading the activity file. Part of the purpose of the **report_switching_activity** command is to help you locate these annotation problems. If possible, you should get complete simulation data. If this is not possible, it can be useful to use the *-average_activity* report to get average toggle rates in the design for the purpose of patching these annotation problems using the **set_switching_activity** command.

EXAMPLES

The following examples report switching accuracy for the current default in the pt_shell.

In the following example, the **report_switching_activity** command is used after the **read_saif** command.

```
pt_shell> report_switching_activity

****************************************
Report : Switching Activity

Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:29:40 2008
****************************************

 Switching Activity Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagatec
-----------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
-----------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------
 Static Probability Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagatec
-----------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
-----------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------
1
```

In the following example, the **report_switching_activity** command is used with the *-list_not_annotated* option to find the nets in the design that do not have switching activity information. Note that the overview report is also printed, and lists several nets as having no activity. These nets are then listed by name below the overview report.

```
pt_shell> report_switching_activity -list_not_annotated

***************************************
Report : Switching Activity
        -list_not_annotated
Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:27:39 2008
***************************************

 Switching Activity Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagated(%
-----------------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
-----------------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------------
 Static Probability Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagated(%
-----------------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
-----------------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------------


List of nonannotated nets :
        a_r[1]
        a_r[0]
        a_r[14]
        a_r[8]
        a_r[13]
        a_r[4]
        a_r[3]
        a_r[10]
        a_r[12]
        a_r[2]
        a_r[5]
        a_r[11]
        a_r[15]
```

```
        a_r[7]
        a_r[9]
        a_r[6]

16 nets with no annotation
1
```

In the following example, the **report_switching_activity** command is used after the **update_power** command. The unannotated nets are now listed as having been propagated.

```
****************************************
Report : Switching Activity

Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:29:16 2008
****************************************

 Switching Activity Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagatec
-----------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        16(0.98%)
-----------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        16(7.11%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------
 Static Probability Overview Statistics for "mac"
-----------------------------------------------------------------------------------------------------
                  From Activity     From         From         From
Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)     Propagatec
-----------------------------------------------------------------------------------------------------
 Nets             1613(99.02%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        16(0.98%)
-----------------------------------------------------------------------------------------------------
Nets Driven by
-----------------------------------------------------------------------------------------------------
Primary Input     67(100.00%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Tri-State         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Black Box         0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Sequential        209(92.89%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        16(7.11%)
Combinational     1337(100.00%)     0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)        0(0.00%)
Memory            0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
Clock Gate        0(0%)             0(0%)        0(0%)        0(0%)        0(0%)           0(0%)
-----------------------------------------------------------------------------------------------------
1
```

In the following example the **report_switching_activity** command is used to get the average toggle rates on the design and subblocks of the design (using the *-hierarchy* flag)

```
pt_shell> report_switching_activity -average -hier

****************************************
Report : Switching Activity
        -average_activity
        -hierarchy
Design : mac
```

```
Version: K-2015.12-DEV-151116
Date   : Tue Nov 17 13:13:22 2015
****************************************

 Switching Activities per period is with respect to time period 9996.000000 ns
 Simulation time for toggle count is 9996 ns
-----------------------------------------------------------------------------------------------------
Object                            Toggle Count       Toggle Rate      Glitch Count   Glitch Rate      #
                                  Per Net            Per Period       Per Net        Per Period       I
                                                     Per Net                         Per Net
-----------------------------------------------------------------------------------------------------
mac                               198.74             0.02             0              0                1
mult_21                           141.16             0.014            0              0                9
mult_21/U1/U9720                  185.49             0.019            0              0                2
add_23                            245.41             0.025            0              0                3
-----------------------------------------------------------------------------------------------------
1
```

In the following example the **report_switching_activity** command is used to get the average toggle rates per clock period on the design and subblocks of the design :

```
pt_shell> report_switching_activity -average -hier -base_clock clk

****************************************
Report : Switching Activity
         -average_activity
         -hierarchy
Design : mac
Version: K-2015.12-DEV-151116
Date   : Tue Nov 17 13:13:47 2015
****************************************

 Switching Activities per period is with respect to Clock 'clk' with period 12.000000 ns
 Simulation time for toggle count is 9996 ns
-----------------------------------------------------------------------------------------------------
Object                            Toggle Count       Toggle Rate      Glitch Count   Glitch Rate      #
                                  Per Net            Per Period       Per Net        Per Period       I
                                                     Per Net                         Per Net
-----------------------------------------------------------------------------------------------------
mac                               198.74             0.24             0              0                1
mult_21                           141.16             0.17             0              0                9
mult_21/U1/U9720                  185.49             0.22             0              0                2
add_23                            245.41             0.29             0              0                3
-----------------------------------------------------------------------------------------------------
1
```

In the following example, the **report_switching_activity** command is used with the *-coverage* flag to determine whether or not the simulation testbench from which the SAIF file was derived did a good job in exercising each part of the design. In this example, the simulation did a poor job in exercising the counter_high_bits subblock.

```
pt_shell> report_switching_activity -coverage -hierarchy

****************************************
Report : Switching Activity
         -coverage -hierarchy
Design : counter
Version: V-2004.06-Alpha1
Date   : Mon Mar 29 11:09:34 2004
****************************************
 Switching Activity Coverage for design "counter"
-----------------------------------------------------------------------------------------------------
 Coverage is defined as the percent of nets with at least 1 toggle
-----------------------------------------------------------------------------------------------------
Block                        % Nets Covered    # Nets Covered    Total Nets
```

```
                                                                    Counted
-----------------------------------------------------------------------------------------------------------
counter                        76                76                100
counter_low_bits              100                44                44
counter_high_bits              25                11                44
reset_block                    50                 3                6
overflow_detect                50                 3                6
-----------------------------------------------------------------------------------------------------------
```

SEE ALSO

```
read_saif(2)
reset_switching_activity(2)
set_switching_activity(2)
get_switching_activity(2)
set_rtl_to_gate_name(2)
update_power(2)
report_power(2)
```

report_taps

Reports information on tap points.

SYNTAX

```
string report_taps
   [taps]
   [-location]
   [-parasitics]
   [-package]
   [-nosplit]
   [-result result_name]
   [-static_current]
```

Data Types

```
list taps
string result_name
```

ARGUMENTS

taps

Specifies a list of taps to be reported. Items within the list can be name patterns or collections of taps. The default is to report information for all of the taps.

-location

Shows the location information for each tap. This information includes the tap type, the associated supply net, and the X, Y and layer coordinates of the tap. By default, the command reports location information in the output report.

-parasitics

Reports the parasitics associated with each tap. This report includes the lumped RLC model type, the resistance value, the capacitance value, and the inductance value for each tap. See the **set_tap_parasitics** man page for further details on tap parasitics.

-package

Reports the SPICE package model and its connection with taps. The report shows the top subcircuit of the SPICE package model and the connectivity between its ports and taps.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents

line-splitting and facilitates writing software to extract information from the report output.

-result *result_name*

Specifies which result context is used to process names or patterns in the tap parameter. The *result_name* can be either *rail* and *integrity*, which represents rail and integrity result context, respectively. See the **tap_contexts** man page for a detailed description of tap contexts. If the *-result result_name* option is specified without a pattern parameter, the command collects all taps from the specified result context. Otherwise, the command looks up and collects the tap names in the tap parameter in the specified result context.

-static_current

Report static current for each tap point. The report is only available from static rail analysis.

DESCRIPTION

The **report_taps** command displays information about the tap points in the current session context, current rail result context or current integrity result context. By default, the command generates a report that includes all current tap points in the session context. Options can be used to limit the report to a subset of the taps. The report contains the following information about each tap:

- Tap name
- Type (auto_pg, top_pg, cell_pg, absolute_coord, cell_coord, libcell_pg and libcell_coord)
- Associated net
- Absolute {X Y} coordinate
- Layer mask name and number from the Milkyway technology file (.tf)
- Name of the object on which the tap was inserted
- Context of the tap point (session, rail and integrity)
- Validity of the tap point location
- Associated SPICE package models

Many of the items included in the **report_taps** output are also accessible as application attributes of the *tap* object class. You can find the list of system attributes for taps with the **list_attributes -application -class tap** command. The **rail_attributes** man page provides definitions of the tap attributes.

EXAMPLES

The following example shows how to report all automatically inserted supply (power or ground) pin taps, then look up the supply net and checking status of a specific tap using attributes, and finally list all the tap object class's attributes:

```
pr_shell> report_taps [get_taps -filter type==auto_pg]

Name      Type       Net    Point             Layer        Object Name     Context      Is_valid
```

```
-------------------------------------------------------------------------
TAP_1     auto_pg    VDD    (123.534, 81.356) (3) M2       /MyTop/VDD      session      true
TAP_2     auto_pg    VSS    (652.100 899.505) (1) M1       /MyTop/VSS      session      true

pr_shell> get_attribute -class tap TAP_1 supply_net
{"VDD"}
pr_shell> get_attribute -class tap TAP_1 is_valid
true

pr_shell> list_attributes -class tap -app

****************************************
Report : List of Attribute Definitions
Version: E-2010.12
Date   : Wed Sep  8 16:38:27 2010
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from design/library (for user-defined)

Attribute Name            Object     Type       Properties  Constraints
-------------------------------------------------------------------------------
full_name                 tap        string     A
is_valid                  tap        string     A
layer                     tap        int        A
parasitics                tap        string     A
position                  tap        string     A
supply_net                tap        collection A
type                      tap        string     A
object_name               tap        string     A
context                   tap        string     A
```

The following example reports taps in the rail result context:

```
pr_shell> report_taps -result rail

Name      Type       Net    Point             Layer        Object Name     Context
-------------------------------------------------------------------------
rail_1    auto_pg    VDD    (100.000 200.000) (3) M2       /MyTop/VDD      rail
rail_2    auto_pg    VSS    (300.230 302.180) (1) M1       /MyTop/VSS      rail
```

The following example reports the package model subcircuit PACKAGE:

```
pr_shell> report_taps -package


Package spice subcircuit: PACKAGE
Package ports           : BC1 BC2 (*)BC3 (*)BC4
(*) floating package ports

Name      Type           Net            Port
-------------------------------------------------------------------------------
BC1       absolute_coord VDD            BC1
BC2       absolute_coord VDD            BC2
```

The report shows BC1 and BC2 have taps created properly but BC3 and BC4 are not connected to any net.

The following example reports taps in the integrity result context:

```
pr_shell> report_taps -result integrity
```

```
Name      Type       Net    Point             Layer        Object Name     Context
-----------------------------------------------------------------------------
int_1     auto_pg    VDD    ( 10.000  20.000) (5) M3                      integrity
int_2     auto_pg    VSS    (110.220 900.188) (3) M2                      integrity
int_3     auto_pg    VSS    (220.220 900.188) (3) M2                      integrity
```

SEE ALSO

check_taps(2)
create_taps(2)
current_integrity_result(2)
current_rail_result(2)
get_taps(2)
rail_attributes(2)
remove_taps(2)
report_integrity_result(2)
report_rail_result(2)
set_tap_parasitics(2)
tap_contexts(2)
write_taps(2)

report_threshold_voltage_group

Reports the number, leakage power, and percentage of cells for each threshold voltage group in the design, including the user-specified low threshold voltage groups.

SYNTAX

```
string report_threshold_voltage_group
   [-lvth_groups groups]
   [-nosplit]
   [-verbose]
   [-pattern_priority pattern_list]
   [-attribute attribute_name_for_cell_pattern]
   [-area]
   cell_list_or_current_design
```

Data Types

groups	list
pattern_list	list
attribute_name_for_cell_pattern	string
cell_list_or_current_design	list

ARGUMENTS

-lvth_groups *groups*

Specifies the list of threshold voltage groups to considered as low threshold voltage groups. The value of the *groups* argument is a list of threshold voltage group identifiers. Each identifier is a nonempty string that consists of alphanumeric characters and the underscore character ("_"). If the option is not specified, the low threshold voltage groups are not reported.

-nosplit

Prevents line splitting when the information exceeds the specified column width. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line beginning with the correct column.

-verbose

Lists all the cells belonging to each threshold voltage group. If the option is not specified, the default summary report shows only the number and percentage of cells.

-pattern_priority *pattern_list*

Specifies the list of patterns of cell name (or its attribute value) to process for generating the threshold voltage group leakage report.

-attribute ***attribute_name_for_cell_pattern***

Specifies the name of the cell attribute where the value is used to match the name patterns supplied with **-pattern_priority** option.

-area

Lists all the cell area belonging to each threshold voltage group. If the option is specified it reports the cell area according to the threshold voltage groups.

cell_list_or_current_design

Specifies a list of cells to report as threshold voltage groups. If the option is not specified, the report is generated for the current design.

DESCRIPTION

Use the **report_threshold_voltage_group** command to report the threshold voltage group for the given list of cells or design (*cell_list_or_current_design*). If no list is provided, a report is generated for the current design. This command identifies the threshold voltage groups to which the cells can be categorized depending on the library-level *default_threshold_voltage_group* attribute or the cell-level **threshold_voltage_group** attribute. The cell-level attribute takes precedence over the library-level attribute in the grouping process.

When you use the **report_threshold_voltage_group -leakage** command, it also reports the leakage power according to the threshold voltage groups. Specify the **update_power** command before the **report_threshold_voltage_group** command. Otherwise, during the leakage-only mode, the tool runs the **update_power** command within the **report_threshold_voltage_group** command.

EXAMPLES

In the following example, library-level attribute is set and then the **report_threshold_voltage_group -lvth_groups** command is specified. This command groups LVT and SVT as low-threshold voltage groups.

```
pt_shell> define_user_attribute -type string -class lib default_threshold_voltage_group
pt_shell> set_user_attribute [get_libs agl90g_od_svt_ss] default_threshold_voltage_group SVT
pt_shell> set_user_attribute [get_libs agl90g_od_lvt_ss] default_threshold_voltage_group LVT
pt_shell> report_threshold_voltage_group -lvth_groups {lvt svt}
-------------------------------------------------------------------------------------------------
                         Threshold Voltage Group Report

Attributes :
------------
         u -> user-defined power group
         L -> user-defined low Vth group

Power Group         ABCD         LVT(L)       XYZK(L)     SVT
Name                cell (%)     cell (%)     cell (%)    cell (%)  Attribute
-----------------------------------------------------------------------------
io_pad              0 (0.00%)    0 (0.00%)    0 (0.00%)   0 (0.00%)
memory              0 (0.00%)    0 (0.00%)    0 (0.00%)   0 (0.00%)
clock_network       0 (0.00%)    0 (0.00%)    0 (0.00%)   0 (0.00%)
black_box           0 (0.00%)    0 (0.00%)    0 (0.00%)   0 (0.00%)
register            0 (0.00%)    0 (0.00%)    1 (25.00%)  3 (75.00%)
combinational       1 (3.33%)   13 (43.33%)   0 (0.00%)  16 (53.33%)
```

```
sequential          0 (0.00%)    4 (100.00%)  0 (0.00%)   0 (0.00%)
--------------------------------------------------------------------------------
Total               1 (2.63%)   17 (44.74%)   1 (2.63%)  19 (50.00%)
--------------------------------------------------------------------------------
SUMMARY :

          Low Vth cell (%) = 18 (47.37%)
          Total cell (%) =   38 (100.00%)
--------------------------------------------------------------------------------

1
```

SEE ALSO

define_user_attribute(2)
report_power(2)
set_user_attribute(2)

report_timing

Reports the timing paths in the design that have the worst slack.

SYNTAX

```
string report_timing
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-exclude exclude_list
      | -rise_exclude rise_exclude_list
      | -fall_exclude fall_exclude_list]
   [-delay_type delay_type]
   [-nworst paths_per_endpoint]
   [-max_paths max_path_count]
   [-group group_name]
   [-unique_pins]
   [-slack_greater_than minimum_slack]
   [-slack_lesser_than  maximum_slack]
   [-ignore_register_feedback feedback_slack_cutoff]
   [-report_ignored_register_feedback]
   [-include_hierarchical_pins]
   [-trace_latch_borrow]
   [-trace_latch_forward]
   [-pba_mode none | path | exhaustive]
   [-normalized_slack]
   [-start_end_pair]
   [-cover_design]
   [-dont_merge_duplicates]
   [-pre_commands  pre_command_string]
   [-post_commands post_command_string]
   [-path_type format]
   [-input_pins]
   [-nets]
   [-nosplit]
   [-transition_time]
   [-capacitance]
   [-significant_digits digits]
   [-crosstalk_delta]
   [-derate]
   [-variation]
   [-exceptions dominant | overridden | all]
   [-voltage]
   [-supply_net_group]
   [-sort_by group | slack]
   [-tag_paths_filtered_by_pba tag_name]
   [path_collection]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
exclude_list	list
rise_exclude_list	list
fall_exclude_list	list
delay_type	string
paths_per_endpoint	integer
max_path_count	integer
group_name	list
minimum_slack	float
maximum_slack	float
feedback_slack_cutoff	float
pre_command_string	string
post_command_string	string
format	string
digits	integer
tag_name	string
path_collection	collection

ARGUMENTS

-from ***from_list***

Reports only paths that start from the specified list of pins, ports, nets, or cell instances; or from startpoints clocked by the named clocks. Valid startpoints are clock pins of sequential devices and input pins of the design.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must start with a rising transition. If you specify a clock object, the option selects startpoints clocked by the named clock and launched by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from ***fall_from_list***

Same as the **-rise_from** option, except that the path must start with a falling transition.

-to ***to_list***

Reports only paths that end at the specified list of pins, ports, nets, or cell instances; or at endpoints clocked by the named clocks. Valid endpoints are data input pins of sequential devices and output pins of the design.

Due to the infrastructure of the timing analyzer, the **-to** option runs more efficiently than the **-from** option. As a result, endpoint-oriented reporting flows (based on using the **-to** option) outperform equivalent startpoint-oriented reporting flows (based on using the **-from** option).

-rise_to ***rise_to_list***

Same as the **-to** option, except that the path data must end with a rising transition. If you specify a clock object, the option selects endpoints clocked by the named clock and captured by a rising edge of the clock at

the clock source, taking into account any logical inversions along the clock path.

-fall_to ***fall_to_list***

Same as the **-rise_to** option, except that the path data must end with a falling transition.

-through ***through_list***

Reports only paths in which the data goes through the named pins, ports, cell instances, or nets. You can use this option multiple times in a command.

When a single **-through** list contains multiple objects, the data path can go through any one of the listed objects. When you use multiple **-through** options, the data path must meet each of the **-through** options in sequence.

For example, the following command reports a path that starts at A1, then passes through either B1 or B2, then passes through C1, and ends at D1.

```
pt_shell> report_timing -from A1 -through {B1 B2} -through C1 -to D1
```

If advanced analysis through transparent latches is enabled (**timing_enable_through_paths** set to **true**), and the pins specified in the last through switch are latch loop breakers, then the timing paths reported are constrained by the worst of the closing edges at the latch or by a required value computed using logic downstream of the latch loop breaker. The latter constraint is called "time required through endpoint" in the timing report, and the latch is said to be constrained by "downstream required."

-rise_through ***rise_through_list***

Same as the **-through** option, except that it applies only to paths with a rising transition at the specified objects.

-fall_through ***fall_through_list***

Same as the **-through** option, except that it applies only to paths with a falling transition at the specified objects.

-exclude ***exclude_list***

Excludes all paths in which the data goes from, through, or to the specified list of pins, ports, nets, or cell instances; or excludes all paths belonging to the specified list of path tag sets. It checks only data paths (not clock paths) for excluded objects.

Using the **-exclude** option can result in significantly longer runtimes. To reduce the runtime impact, use this option together with other topological restriction options such as **-from** and **-through**.

The **-exclude** option does not check borrowing paths considered by the **-trace_latch_borrow** option.

-rise_exclude ***rise_exclude_list***

Same as the **-exclude** option, but applies only to rising transitions at the named pins, ports, nets, or cell instances.

-fall_exclude ***fall_exclude_list***

Same as the **-exclude** option, but applies only to rising transitions at the named objects.

You can use no more than one of the **-exclude**, **-rise_exclude**, or **-fall_exclude** options in a command.

-delay_type ***delay_type***

Specifies the type of path delay constraint to consider for finding and sorting paths with the worst slack:

- **max** (default) -- max delay (setup constraint)
- **min** -- min delay (hold constraint)
- **min_max** -- both min and max delay (reported min first, max second)
- **max_rise** -- max delay, rising at data path endpoint
- **max_fall** -- max delay, falling at data path endpoint
- **min_rise** -- min delay, rising at data path endpoint
- **min_fall** -- min delay, falling at data path endpoint

-nworst *paths_per_endpoint*

Reports up to the specified number of worst paths per endpoint. The default is 1. A larger value results in a larger report and more runtime. Allowed values are 1 to 2000000.

If you set **-nworst** to any number greater than 1, the command automatically does the following:

- Implicitly sets **-max_paths** equal to the **-nworst** setting if the **-max_paths** option is not used in the command, so that at least one set of multiple paths to an endpoint can be reported
- Implicitly sets **-slack_lesser_than 0.0** if the **-slack_lesser_than** and **-slack_greater_than** options are not used in the command, so that only violating paths are reported

-max_paths *max_path_count*

Reports up to the specified maximum total number of paths among all path groups. The default is equal to the **-nworst** setting, or 1 if the **-nworst** option is not used. Allowed values are 1 to 2000000.

If you set **-max_paths** to any number greater than 1, the command implicitly sets **-slack_lesser_than 0.0** if the **-slack_lesser_than** and **-slack_greater_than** options are not used in the command, so that only violating paths are reported.

The **-nworst** option controls the maximum number of paths reported *per endpoint*, whereas the **-max_paths** option controls the *overall total maximum number* of paths reported. The **-max_paths** setting should be at least as large as the **-nworst** setting, so by default the tool implicitly sets **-max_paths** to the same value as **-nworst** if you do not use the **-max_paths** option in the command.

-group *group_name*

Reports only paths that belong to the specified path groups. Path groups are created by the **create_clock** and **group_path** commands. Without the **-group** option, the **report_timing** command reports paths without considering path groups. For example, the **report_timing** command alone reports the single worst path in the design, whereas **report_timing -group [get_path_groups]** reports the single worst path *in each path group*.

-unique_pins

Reports only the single worst timing path through any given sequence of pins. No other paths are reported for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not reported for that sequence of pins.

For non-unate logic such as XOR gates, using this option greatly reduces the number of paths reported because of the large number of possible rising/falling edge combinations through each sequence of pins.

Using this option can require longer runtimes when used with the **-nworst** option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is reported and counted

toward the total number of requested paths.

-slack_greater_than ***minimum_slack***

Reports only paths with slack greater than the specified *minimum_slack* value; these paths have a negative slack better than the specified *minimum_slack* value (or a positive slack that is farther from causing a violation). This option is intended to be used with the **-slack_lesser_than** option to report paths within a specific range of slack.

-slack_lesser_than ***maximum_slack***

Reports only paths with slack less than the specified *maximum_slack* value; these paths have a negative slack worse than the specified *maximum_slack* value (or a positive slack that is closer to causing a violation).

By default, the **report_timing** command reports the single path with the worst slack; or if the **-nworst** or **-max_paths** option is set to a value greater than 1, the command considers all paths with negative slack, as if **-slack_lesser_than** is set to 0.0.

The **-slack_lesser_than** and **-slack_greater_than** options act as filters to restrict the paths that are reported, so the number of paths reported can be fewer than the number specified with the **-nworst** and **-max_paths** options. If no paths meet the slack criteria, no paths are reported.

-ignore_register_feedback ***feedback_slack_cutoff***

Ignores noninverting timing loops that start and end at the same register pin that holds a value. To be ignored, the data-to-output arc and the output-to-data path must be either both inverting or both noninverting. This option applies to minimum delay as well as maximum delay constraints. Paths are ignored only if they have a slack less than the specified *feedback_slack_cutoff* value.

This option acts as a filter to restrict the paths that are reported, so the number of paths reported might be less than the number specified with the **-nworst** and **-max_paths** options.

-report_ignored_register_feedback

Displays a list of ignored paths when you use the **-ignore_register_feedback** option.

-include_hierarchical_pins

Causes the timing path report to show points for the hierarchical pins crossed, as well as all leaf pins in the path. The hierarchical pins are shown only for documentation purposes; they are reported as having zero incremental delay and do not affect the timing results.

-trace_latch_borrow

Controls the type of report generated for a path that starts at a transparent latch. If the path startpoint borrows from the previous path segment, using this option causes the report to show the set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. Each path segment is reported separately, showing the time borrowed and lent and the endpoints of the path segment.

Without the **-trace_latch_borrow** option, the command does not include upstream borrowing latches when in default latch analysis mode or borrowing latch loop breakers when in advanced latch analysis mode (**timing_enable_through_paths** variable set to true) as path segments in the timing report.

-trace_latch_forward

Controls the type of report generated for a path that ends at a transparent latch D pin. If advanced analysis through transparent latches is enabled (**timing_enable_through_paths** is set to true) and this transparent latch D pin is constrained by a downstream required time, this option causes the report to show the paths in the fanout of this D pin that led to the downstream required constraint. These paths can include multiple path segments whereby the constraint is the result of propagating through more than one latch constrained by the

downstream required time. In this case, each path segment is reported separately, showing the downstream required time for that path segment.

Without the **-trace_latch_forward** option, the command does not trace the path beyond the specified endpoint.

-pba_mode none | path | exhaustive

Specifies one of the following path-based timing analysis modes:

- **none** (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- **path** - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths. To control the ordering of the paths (either by original graph-based slack or recalculated path-based slack), set the **pba_path_mode_sort_by_gba_slack** variable.
- **exhaustive** - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode. You cannot use the **exhaustive** mode together with the **-start_end_pair**, **-cover_design**, or *path_collection* options.

In ordinary graph-based (default) timing analysis, the tool considers both the worst arrival time and worst slew among all signals feeding into a path, even when the worst arrival and worst slew come from different signals.

In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

-normalized_slack

Gathers, sorts, and reports paths using normalized slack instead of absolute slack. Normalized slack is the absolute slack divided by an idealized maximum allowable propagation delay (typically the clock period). To use this option, the timing update must be performed while the **timing_enable_normalized_slack** variable is set to **true**.

-start_end_pair

Reports the worst path for each startpoint-endpoint pair in the design, including all violating startpoint-endpoint pairs by default, or all startpoint-endpoint pairs with slack worse than the value specified by the **-slack_lesser_than** option if that option is used. The number of paths reported is limited to 2000000.

The **-start_end_pair** option overrides the default behavior of reporting only the single worst path in the design. Instead, the command reports the worst timing path of each violating startpoint-endpoint pair, which can result in a very large report and long runtimes. (Note that the Design Compiler and IC Compiler tools differ from this behavior by respecting the default limits on the number of paths reported and by allowing usage of the **-nworst** and **-max_paths** with **-start_end_pair**.)

To limit the report size and runtime to reasonable amounts, use the **-slack_lesser_than** option and specify a slack limit that is only slightly greater than the worst slack in the design. You can also restrict the scope of the analysis by using the **-from**, **-to**, and similar options. You cannot use the **-start_end_pair** option with the **-nworst**, **-max_paths**, **-unique_pins**, **-slack_greater_than**, or **-ignore_register_feedback** options.

Using the **-start_end_pair** option requires any **-from**, **-rise_from**, or **-fall_from** option to specify only valid timing startpoints, and any **-to**, **-rise_to**, or **-fall_to** option to specify only valid timing endpoints; wildcard characters that specify invalid startpoints or endpoints are not allowed. In other words, the **timing_report_always_use_valid_start_end_points** variable is treated as set to **true**, irrespective of its actual setting.

The **-start_end_pair** option sorts the paths first by endpoint and then by slack, with the paths sharing the same endpoint reported together in order of increasing slack.

-cover_design

Reports the worst path through each violating pin in the design, so that the reported paths cover every violating pin in the design. A pin is deemed to be violating if its slack is less than zero, or less than the value specified with the **-slack_lesser_than** option if that option is used. The slack of a pin is the worst slack among all paths that start at, pass through, or end on that pin. The maximum number of paths reported is 2000000.

The following options cannot be used with **-cover_design**:

```
-nworst
-max_paths
-unique_pins,
-ignore_register_feedback
-report_ignored_register_feedback
-start_end_pair
-pba_mode exhaustive
path_collection
```

-dont_merge_duplicates

Prevents the merging of duplicate paths across multiple scenarios when the PrimeTime tool is invoked with the **-multi_scenario** option.

By default, when the same path is reported in more than one scenario, the tool reports only the single most critical instance of that path in the merged report and shows its associated scenario.

If you use this option, the tool does not merge duplicate instances of the same path, but instead reports all critical instances of the path from all scenarios. Because the number of paths reported might limited by **-nworst**, **-max_paths**, or other command options, the resulting merged report might be more focused on a portion of the design that is critical in multiple scenarios, and less evenly spread out across different portions of design.

-pre_commands *pre_command_string*

Specifies a list of commands, separated by semicolons, to be executed in the worker (slave) context before execution of the **report_timing** command. This option is available only if you invoke the PrimeTime tool with the **-multi_scenario** option. The maximum size of a command is 1000 characters.

-post_commands *post_command_string*

This option is like the **-pre_commands** option, except that the commands are executed after (instead of before) the **report_timing** command.

-path_type *format*

Specifies the types of path information displayed in the timing report, using one of the following format keywords:

- **summary** -- Displays the path startpoint, path endpoint, and slack.
- **end** -- Displays the path endpoint, path delay, required time, and slack.
- **short** -- Displays a report like the default **full** report but omits the intermediate points between the startpoint and endpoint.
- **full** (the default) -- Displays the full data path, including all intermediate points and their incremental and cumulative delay, together with the launch and capture clock arrival times, data required time, and slack.
- **full_clock** -- Displays a **full** report with the addition of the clock path points from the clock source to the

launch and capture points.

- **full_clock_expanded** -- Displays a **full_clock** report with the addition of the clock path points between the primary clock source and its related generated clock source point. This report is different from a **full_clock** report only when the clock is a generated clock.

-input_pins

Shows cell input pins as well as cell output pins in the timing path. As a result, the report shows the incremental net and cell delays separately at each point, instead of combined. By default, the report shows only the cell output pins.

-nets

Shows nets in the timing path. By default, the report does not show nets.

-nosplit

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the alloted space in a column.

-transition_time

Shows the transition time (slew) in the path report for each driver pin and load pin, appearing as an additional column labeled "Trans". By default, the report does not show transition time.

-capacitance

Shows the total capacitance in the path report for each net, appearing as an additional column labeled "Cap". By default, the report does not show capacitance.

The reported value is either the CCS receiver capacitance (the default) or the lumped capacitance, depending on the **report_capacitance_use_ccs_receiver_model** variable setting.

-significant_digits *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, which is 2 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

-crosstalk_delta

Reports the annotated delta delay values at cell input pins in a column labeled "Delta". The **-input_pins** option is automatically selected. The delta values are computed during crosstalk signal integrity analysis, or they can be annotated manually by using the **set_annotated_delay -delta_only** and **set_annotated_transition -delta_only** commands. Note that the **-crosstalk_delta** option only reports previously calculated or annotated delta values; it does not initiate crosstalk analysis. Delta transition times are also shown if you use the **-transition_time** option.

-derate

Shows derating factors in a column labeled "Derate". The default is to not show derating factors. Specifying this option automatically sets the **-input_pins** option, so that different net and cell derating values are reported. When you use the **-path_type full_clock_expanded** option, an additional summary report follows the timing report showing the overall effect of derating the path. Derating factors are always shown with at least two significant digits.

-variation

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit".

POCV analysis is enabled by setting the **timing_pocvm_enable_analysis** variable to **true**.

-exceptions dominant | overridden | all

Reports user-specified timing exceptions that are satisfied per timing path being reported.

Specifying dominant reports the dominant timing exception in the path. It can be one of the following: false path, minimum and maximum delays, and multicycle paths.

Specifying overridden reports all the timing exceptions that were overridden by the dominant timing exception.

The unconstrained paths and their reason is reported for all three options if **timing_report_unconstrained_paths** variable is set to true, otherwise unconstrained paths will not be reported.

To get more information about the exceptions that apply to the path, you can create a path collection with the **get_timing_paths** command and query certain timing path attributes. For details, see the man page for the **get_timing_paths** command.

Note: The additional analysis required per path with the **-exceptions** option can be significant, so you can expect a longer runtime. This option cannot be used with the **-path_type short/end/summary** option.

-voltage

Reports the operating voltage for each path element in a column labeled "Voltage", allowing you to debug voltage levels in multivoltage designs. The displayed voltage is the voltage used in delay calculation, which is typically the voltage of the operating condition, related supply net, related power pin, or pin signal.

-supply_net_group

Reports the supply group or the supply net name for each path element in a column labeled "Supply-Net-Group". If the supply net group name is not defined, the column shows the word "unconnected". If the supply group is not defined and the supply net name is available, then the supply net name is reported in the column along with the letter "n".

-sort_by group | slack

Specifies the sorting of the reported paths, either by **group** (the default) or by **slack**. Sorting by group means ordering the path reports by group, and within each group, by slack. Ordering by slack means ordering the path reports strictly by slack, ignoring group membership.

-tag_paths_filtered_by_pba *tag_name*

When used with the **-pba_mode path** option, the **-tag_paths_filtered_by_pba** option causes tagging of all paths originally found by path-based analysis and later discarded by path-based analysis, using the specified tag name. In future **get_timing_paths** and **report_timing** commands, you can exclude the tagged paths from further analysis by specifying the tag name as an argument to the **-exclude** option.

path_collection

Specifies a collection of timing paths to report, for example, a collection previously created by **set mypaths [get_timing_paths ...]**. When you use this option with the **-pba_mode path** option, the command performs path-based analysis on the paths in the *path_collection* before they are reported. This option is mutually exclusive with options that control the selection of paths to report and is compatible only with the options that control the formatting of the report.

DESCRIPTION

The **report_timing** command reports the timing paths in the current design that have the worst slack. These are the paths that violate the timing constraints by the largest amounts, or paths with positive slack that come closest to causing a timing violation.

Each path has a startpoint and an endpoint. Data is launched by a clock edge at the path startpoint, propagated through combinational logic in the path, and then captured at the path endpoint by another clock edge. The startpoint can be a register clock pin or an input port. The endpoint can be a register data input pin or an output port.

To restrict the scope of the report to only the paths that start, pass through, or end on specific pins, ports, nets, or cell instances, use the **-from**, **-though**, and **-to** options, and the similar transition-specific options (**-rise_from**, **-fall_to**, and so on). To restrict the report to specific clocks, you can use the same options (for example, **-to [get_clocks CLK1]**) or the **-group** option.

By default, the **report_timing** command without options reports the single path in the design with the worst max delay (setup constraint) violation. To consider constraints other than max delay, use the **-delay_type** option. To control the number of paths reported, use the **-nworst** option, which specifies the maximum number of worst paths reported per endpoint; and the **-max_paths** option, which specifies the overall maximum number of paths reported by the command.

To report paths whose slack values fall within a specific range, use the **-slack_lesser_than** and **-slack_greater_than** options in combination with the **-nworst** and **-max_paths** options.

The default timing report shows the point-to-point sequence of cell output pins in the data path, as in the following example:

```
Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
             (rising edge-triggered flip-flop clocked by PCI_CLK)
Endpoint: pad[1] (output port clocked by PCI_CLK)
Path Group: PCI_CLK
Path Type: max

Point                                              Incr       Path
-----------------------------------------------------------------
clock PCI_CLK (rise edge)                         0.000      0.000
clock network delay (propagated)                  1.105      1.105
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)      0.000      1.105 r
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)       0.404 &    1.509 r
U7/ZN (inv0d1)                                    0.066 &    1.575 f
U62/ZN (inv0d1)                                   0.042 &    1.617 r
U63/Z (or02d7)                                    0.269 &    1.887 r
U63ASTipoInst495/Z (bufbdk)                       0.306 &    2.192 r
U63ASTipoInst494/Z (bufbda)                       3.798 &    5.991 r
pad_iopad_1/PAD (pc3b03)                          6.118 H   12.109 r
pad[1] (inout)                                    0.048     12.157 r
data arrival time                                           12.157

clock PCI_CLK (rise edge)                        15.000     15.000
clock network delay (propagated)                  0.000     15.000
output external delay                            -4.000     11.000
data required time                                          11.000
-----------------------------------------------------------------
data required time                                          11.000
data arrival time                                          -12.157
-----------------------------------------------------------------
slack (VIOLATED)                                            -1.157
```

The report shows the incremental delay contributed by each pin-to-pin segment of the path (Incr column) and the cumulative path delay up to each pin in the timing path (Path column). The table also shows the data launch clock arrival time, the data capture clock arrival time, an accounting of arrival time versus required time, and the resulting timing slack for the path. A negative slack indicates a timing violation.

To calculate the delay contributed by each point along the path, the tool considers any back-annotated RC network or SDF delay information. The types of back-annotated data on path elements in the timing path are indicated by a character symbol in the Incr column:

```
Symbol      Annotation
------      ----------
  H         Hybrid annotation
  ^         Ideal network latency annotation
  *         SDF back-annotation
  &         RC network back-annotation
  $         RC pi back-annotation
  +         Lumped RC
  @         HyperScale annotation
none        Wire-load model or none
```

Certain annotations dominate others. For example, SDF takes precedence over back-annotated RC parasitics. The symbol displayed in the Incr column indicates the dominant annotation on the path element.

If the **-input_pins** option is used, the report includes the input pins as well as output pins of each cell in the path. In that case, each pin-to-pin delay spans either a net or cell. Without the **-input_pins** option, the delay shown can span both a net and a cell. If the net and cell have the same dominant annotation, the appropriate symbol is shown; otherwise, the "H" symbol indicates that a hybrid of annotation types exists on the corresponding path segment.

The dominant annotation is not necessarily the actual annotation used to calculate the delay. For example, suppose that a back-annotated RC network exists on a net, but the network calculation fails to converge for the driver cell delay. In that case, you get a warning message and a lumped RC model is used instead. However, in the timing report, the "&" symbol still reports the RC network annotation as the dominant annotation. To see the actual pin-to-pin delay calculation in detail, use the **report_delay_calculation** command.

To control the amount of detail provided in the point-to-point timing report, use the following options:

- **-path_type** -- Specifies the report format: **short**, **full**, **full_clock**, …
- **-input_pins** -- Lists the cell input pins (in addition to output pins) in the path
- **-nets** -- Lists the nets (as well as pins) in the path
- **-transition_time** -- Shows transition time (slew) in a separate column
- **-capacitance** -- Shows capacitance values in a separate column
- **-crosstalk_delta** -- Shows annotated delta delay values in a separate column
- **-derate** -- Shows derating factors in a separate column
- **-variation** -- Shows POCV mean and sensitivity values in separate columns
- **-voltage** -- Shows operating voltages in a separate column

To invoke path-based analysis, use the **-pba_mode** option. In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

EXAMPLES

The following example reports the single path in the design with the worst max-delay (setup) slack:

```
pt_shell> report_timing
****************************************
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -sort_by slack
        ...
****************************************

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                                                Incr       Path
  -----------------------------------------------------------------------
  clock PCI_CLK (rise edge)                           0.000      0.000
  clock network delay (propagated)                    1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)        0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)         0.404 &    1.509 r
  U7/ZN (inv0d1)                                      0.066 &    1.575 f
  U62/ZN (inv0d1)                                     0.042 &    1.617 r
  U63/Z (or02d7)                                      0.269 &    1.887 r
  U63ASTipoInst495/Z (bufbdk)                         0.306 &    2.192 r
  U63ASTipoInst494/Z (bufbda)                         3.798 &    5.991 r
  pad_iopad_1/PAD (pc3b03)                            6.118 H   12.109 r
  pad[1] (inout)                                      0.048     12.157 r
  data arrival time                                             12.157

  clock PCI_CLK (rise edge)                          15.000     15.000
  clock network delay (propagated)                    0.000     15.000
  clock reconvergence pessimism                       0.000     15.000
  output external delay                              -4.000     11.000
  data required time                                            11.000
  -----------------------------------------------------------------------
  data required time                                            11.000
  data arrival time                                            -12.157
  -----------------------------------------------------------------------
  slack (VIOLATED)                                              -1.157
```

The following example shows the cell input pins as well as cell output pins in the path. As a result, the report shows the incremental net and cell delays separately at each point.

```
pt_shell> report_timing -input_pins
  ...

  Point                                                Incr       Path
  -------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                           0.000      0.000
  clock network delay (propagated)                    1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)        0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)         0.404 &    1.509 r
  U7/I (inv0d1)                                       0.008 &    1.517 r
  U7/ZN (inv0d1)                                      0.058 &    1.575 f
  U62/I (inv0d1)                                      0.004 &    1.579 f
  U62/ZN (inv0d1)                                     0.038 &    1.617 r
  U63/A2 (or02d7)                                     0.004 &    1.621 r
  U63/Z (or02d7)                                      0.265 &    1.887 r
  U63ASTipoInst495/I (bufbdk)                         0.054 &    1.941 r
  U63ASTipoInst495/Z (bufbdk)                         0.252 &    2.192 r
  U63ASTipoInst494/I (bufbda)                         2.664 &    4.857 r
```

```
  U63ASTipoInst494/Z (bufbda)                      1.134 &    5.991 r
  pad_iopad_1/OEN (pc3b03)                         1.444 &    7.434 r
  pad_iopad_1/PAD (pc3b03)                         4.675     12.109 r
  pad[1] (inout)                                   0.048     12.157 r
  data arrival time                                          12.157
  ...
```

The following example shows the net names and net fanout as well as cell output pins in the path.

```
pt_shell> report_timing -nets
  ...

  Point                                       Fanout     Incr       Path
  ---------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                              0.000      0.000
  clock network delay (propagated)                       1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)           0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)            0.404 &    1.509 r
  I_ORCA_TOP/I_PCI_CORE/pad_en (net)             2
  U7/ZN (inv0d1)                                         0.066 &    1.575 f
  n43 (net)                                      1
  U62/ZN (inv0d1)                                        0.042 &    1.617 r
  n8 (net)                                       1
  U63/Z (or02d7)                                         0.269 &    1.887 r
  n134 (net)                                     8
  U63ASTipoInst495/Z (bufbdk)                            0.306 &    2.192 r
  n134ASTipoNet251 (net)                         3
  U63ASTipoInst494/Z (bufbda)                            3.798 &    5.991 r
  n134ASTipoNet250 (net)                         1
  pad_iopad_1/PAD (pc3b03)                               6.118 H   12.109 r
  pad[1] (net)                                   1
  pad[1] (inout)                                         0.048     12.157 r
  data arrival time                                                12.157
  ...
```

The following example reports the worst path that passes through a specified cell and ends at a specified port. The pins of the "through" cell are marked with arrow notation "<-" in the report.

```
pt_shell > report_timing -through [get_cells U63] -to [get_ports pad[1]]
  ...

  Point                                            Incr       Path
  ---------------------------------------------------------------------
  clock PCI_CLK (rise edge)                       0.000      0.000
  clock network delay (propagated)                1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)    0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)     0.404 &    1.509 r
  U7/ZN (inv0d1)                                  0.066 &    1.575 f
  U62/ZN (inv0d1)                                 0.042 &    1.617 r
  U63/A2 (or02d7) <-                              0.004 &    1.621 r
  U63/Z (or02d7) <-                               0.265 &    1.887 r
  U63ASTipoInst495/Z (bufbdk)                     0.306 &    2.192 r
  U63ASTipoInst494/Z (bufbda)                     3.798 &    5.991 r
  pad_iopad_1/PAD (pc3b03)                        6.118 H   12.109 r
  pad[1] (inout)                                  0.048     12.157 r
  data arrival time                                         12.157

  clock PCI_CLK (rise edge)                      15.000     15.000
  clock network delay (propagated)                0.000     15.000
  clock reconvergence pessimism                   0.000     15.000
  output external delay                          -4.000     11.000
  data required time                                        11.000
  ---------------------------------------------------------------------
```

```
  data required time                                          11.000
  data arrival time                                          -12.157
  -------------------------------------------------------------------
  slack (VIOLATED)                                            -1.157
```

The following example reports the endpoint path delay, required time, clock reconvergence pessimism removal (if applicable), and slack for the five worst max-delay paths:

```
pt_shell> report_timing -path_type end -max_paths 5
****************************************
Report : timing
 -path_type end
 -delay_type max
 -slack_lesser_than 0.000
 -max_paths 5
 -sort_by slack
        ...
****************************************

Endpoint                        Path Delay     Path Required    CRP    Slack
-------------------------------------------------------------------------------
pad[1] (inout)                    12.157 r       11.000        0.000   -1.157
I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]/D (sdnrb1)
                                  12.004 f       10.547        0.470   -0.987
pad[14] (inout)                   11.874 r       11.000        0.000   -0.874
pad[13] (inout)                   11.874 r       11.000        0.000   -0.874
I_ORCA_TOP/I_BLENDER/s4_op1_reg[31]/D (sdnrb1)
                                  11.643 f       10.520        0.282   -0.841
```

The following example includes capacitance and transition time in the report:

```
pt_shell> report_timing -capacitance -transition_time
  ...

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                                  Cap      Trans       Incr       Path
  -------------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                                  0.000      0.000
  clock network delay (propagated)                           1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)
                                                  0.363      0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)
                                         0.008      0.140      0.404 &    1.509 r
  U7/ZN (inv0d1)                         0.004      0.089      0.066 &    1.575 f
  U62/ZN (inv0d1)                        0.004      0.071      0.042 &    1.617 r
  U63/Z (or02d7)                         0.060      0.207      0.269 &    1.887 r
  U63ASTipoInst495/Z (bufbdk)            3.458      0.136      0.306 &    2.192 r
  U63ASTipoInst494/Z (bufbda)            1.725      0.524      3.798 &    5.991 r
  pad_iopad_1/PAD (pc3b03)               8.586      1.272      6.118 H   12.109 r
  pad[1] (inout)                                    1.272      0.048     12.157 r
  data arrival time                                                      12.157

  ...
```

The following example reports nets and input pins, and includes capacitance and transition time in the report:

```
pt_shell> report_timing -nets -input_pins -capacitance -transition_time
  ...

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                        Fanout    Cap     Trans      Incr       Path
  ---------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                                0.000      0.000
  clock network delay (propagated)                         1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)
                                                 0.363     0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)    0.140     0.404 &    1.509 r
  I_ORCA_TOP/I_PCI_CORE/pad_en (net)
                                 2    0.008
  U7/I (inv0d1)                                  0.142     0.008 &    1.517 r
  U7/ZN (inv0d1)                                 0.089     0.058 &    1.575 f
  n43 (net)                      1    0.004
  U62/I (inv0d1)                                 0.089     0.004 &    1.579 f
  U62/ZN (inv0d1)                                0.071     0.038 &    1.617 r
  n8 (net)                       1    0.004
  U63/A2 (or02d7)                                0.072     0.004 &    1.621 r
  U63/Z (or02d7)                                 0.207     0.265 &    1.887 r
  n134 (net)                     8    0.060
  U63ASTipoInst495/I (bufbdk)                    0.276     0.054 &    1.941 r
  U63ASTipoInst495/Z (bufbdk)                    0.136     0.252 &    2.192 r
  n134ASTipoNet251 (net)         3    3.458
  U63ASTipoInst494/I (bufbda)                    8.904     2.664 &    4.857 r
  U63ASTipoInst494/Z (bufbda)                    0.524     1.134 &    5.991 r
  n134ASTipoNet250 (net)         1    1.725
  pad_iopad_1/OEN (pc3b03)                       4.964     1.444 &    7.434 r
  pad_iopad_1/PAD (pc3b03)                       1.272     4.675     12.109 r
  pad[1] (net)                   1    8.586
  pad[1] (inout)                                 1.272     0.048     12.157 r
  data arrival time                                                  12.157
  ...
```

The following example shows an unconstrained path. The command first looks for constrained paths; if none are found, it reports unconstrained paths.

```
pt_shell> set_app_var timing_report_unconstrained_paths true
true
pt_shell> report_timing -to [get_ports pad[5]]
 ...
  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop)
  Endpoint: pad[5] (output port)
  Path Group: (none)
  Path Type: max

  Point                                                Incr       Path
  ---------------------------------------------------------------------
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)       0.000      0.000 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)        0.404 &    0.404 r
  U7/ZN (inv0d1)                                     0.069 &    0.473 f
  U62/ZN (inv0d1)                                    0.043 &    0.515 r
  U63/Z (or02d7)                                     0.269 &    0.785 r
  U63ASTipoInst499/Z (bufbda)                        0.243 &    1.027 r
  U63ASTipoInst498/Z (bufbda)                        2.205 &    3.232 r
```

```
  pad_iopad_5/PAD (pc3b03)                          6.121 H    9.354 r
  pad[5] (inout)                                    0.048      9.402 r
  data arrival time                                            9.402
  -------------------------------------------------------------------------
  (Path is unconstrained)
```

The following example shows a timing report with the **-derate** and **-path_type full_clock_expanded** options. The **-derate** option shows the applicable derating factors in a separate column. The **-path_type full_clock_expanded** option reports the clock path points from the clock source to the launch and capture points, including points from the primary clock and its related generated clock; and also displays a derating summary report at the end.

```
pt_shell> set_timng_derate -late 1.04
1
pt_shell> report_timing -derate -path_type full_clock_expanded
 ...
  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
               (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max

  Point                          Derate    Incr       Path
  -----------------------------------------------------------
  clock PCI_CLK (rise edge)                0.000      0.000
  clock source latency                     0.000      0.000
  pclk (in)                                0.000      0.000 r
  pclk_iopad/PAD (pc3d01)        1.040     0.048      0.048 r
  pclk_iopad/CIN (pc3d01)        1.040     0.792 &    0.841 r
  I_CLOCK_GEN/I_PLL_PCI/REF_CLK (PLL)
                                 1.040     0.034 &    0.874 r
  I_CLOCK_GEN/I_PLL_PCI/CLK (PLL)
                                 1.040    -1.211 *   -0.337 r
  I_CLOCK_GEN/bufbdfG1B1I1_1/I (bufbdf)
                                 1.040     0.008 &   -0.329 r
  I_CLOCK_GEN/bufbdfG1B1I1_1/Z (bufbdf)
                                 1.040     0.205 &   -0.124 r
  I_CLOCK_GEN/U21/I0 (mx02d2)    1.040     0.004 &   -0.120 r
  I_CLOCK_GEN/U21/Z (mx02d2)     1.040     0.187 &    0.067 r
  I_CLOCK_GEN/bufbdfG2B1I1_2/I (bufbdf)
                                 1.040     0.008 &    0.075 r

  ...

  pad_iopad_1/PAD (pc3b03)       1.040     4.862     12.694 r
  pad[1] (inout)                 1.040     0.050     12.745 r
  data arrival time                                  12.745

  clock PCI_CLK (rise edge)                15.000     15.000
  clock network delay (propagated)          0.000     15.000
  clock reconvergence pessimism             0.000     15.000
  output external delay                    -4.000     11.000
  data required time                                  11.000
  -----------------------------------------------------------
  data required time                                  11.000
  data arrival time                                  -12.745
  -----------------------------------------------------------
  slack (VIOLATED)                                    -1.745

  Derate Summary Report
  ---------------------------------------------------
  total derate : required time             0.000
  total derate : arrival time             -0.587
```

```
  ---------------------------------------------------
  total derate : slack                    0.587

  slack (with derating applied) (VIOLATED)
                                         -1.745
  clock reconvergence pessimism (due to derating)
                                          0.000
  ---------------------------------------------------
  slack (with no derating) (VIOLATED)    -1.157
```

The following example includes crosstalk delta delays and transition times:

```
pt_shell> report_timing -transition_time -capacitance -nets \
          -crosstalk_delta -significant_digits 4
****************************************
Report : timing
        -path_type full
        -delay max
        -input_pins
        -nets
        -max_paths 1
        -transition_time
        -capacitance
        -crosstalk_delta
Design : TestBH
****************************************

  Startpoint: FirstReg (rising edge-triggered flip-flop clocked by myclock2)
  Endpoint: SecondReg (rising edge-triggered flip-flop clocked by myclock3)
  Path Group: myclock3
  Path Type: max

  Point                       Fanout    Cap    DTrans   Trans    Delta     Incr       Path
  ------------------------------------------------------------------------------------------
  clock myclock2 (rise edge)                                              2.0000     2.0000
  clock network delay (propagated)                                        0.0000     2.0000
  FirstReg/CP (FD1Q)                                  0.1634              0.0000     2.0000 r
  FirstReg/Q (FD1Q)                                   0.4030              0.4286 &   2.4286 f
  OutFirstReg (net)             1   0.0996
  InstSimple/InC (SimpleComb)                         0.0000              0.0000     2.4286 f
  InstSimple/InC (net)
  InstSimple/InterNand/B (ND2)                0.1435  0.5467   0.1030     0.1092 &   2.5377 f
  InstSimple/InterNand/Z (ND2)                        0.7001              0.3598 &   2.8976 r
  InstSimple/OutS (net)         1   0.1001
  InstSimple/OutS (SimpleComb)                        0.0000              0.0000     2.8976 r
  Reg (net)
  SecondReg/D (FD1Q)                          0.1540  0.8542   0.2562     0.2615 &   3.1590 r
  data arrival time                                                                  3.1590

  clock myclock3 (rise edge)                                              3.0000     3.0000
  clock network delay (propagated)                                        0.0000     3.0000
  SecondReg/CP (FD1Q)                                                                3.0000 r
  library setup time                                                     -0.2251     2.7749
  data required time                                                                 2.7749
  ------------------------------------------------------------------------------------------
  data required time                                                                 2.7749
  data arrival time                                                                 -3.1590
  ------------------------------------------------------------------------------------------
  slack (VIOLATED)                                                                  -0.3841
```

The following example shows a timing report with the **-exceptions all** option:

```
pt_shell> report_timing -from ff1/CP -to ff3/D -exceptions all
```

```
****************************************
Report : timing
        -path_type full
        -delay max
        -max_paths 1
 ...
****************************************

  Startpoint: ff1 (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ff3 (rising edge-triggered flip-flop clocked by CLK)
  Path Group: CLK
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  ff1/CP (FD1)                             0.00       0.00 r
  ff1/Q (FD1)                              1.43       1.43 f
  inv1/Z (IV)                              0.54       1.97 r
  and1/Z (AN2)                             0.80       2.78 r
  inv3/Z (IV)                              0.25       3.02 f
  ff3/D (FD1)                              0.00       3.02 f
  data arrival time                                   3.02

  max_delay                                3.00       3.00
  library setup time                      -0.80       2.20
  data required time                                  2.20
  -----------------------------------------------------------
  data required time                                  2.20
  data arrival time                                  -3.02
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.83


The dominant exceptions are:
From          Through        To             Setup                    Hold
--------------------------------------------------------------------------

*             inv3/Z         *              max=3

The overridden exceptions are:
From          Through        To             Setup                    Hold
--------------------------------------------------------------------------

*             inv1/Z         *              max=5

1
```

To report the worst path in the design using path-based analysis:

```
pt_shell> report_timing -pba_mode exhaustive
 ...
```

To report all endpoints in the design which fail after considering path-based analysis:

```
pt_shell> report_timing -pba_mode exhaustive -slack_lesser_than 0 -max_paths 1000
 ...
```

SEE ALSO

get_timing_paths(2)

```
report_constraint(2)
report_delay_calculation(2)
report_supply_group(2)
set_case_analysis(2)
report_default_significant_digits(3)
si_enable_analysis(3)
timing_enable_through_paths(3)
timing_report_always_use_valid_start_end_points(3)
timing_report_fixed_width_columns_on_left(3)
timing_report_unconstrained_paths(3)
```

report_timing_derate

Reports derating previously set by the **set_timing_derate** command.

SYNTAX

```
status report_timing_derate
   [-include_inherited]
   [-variation]
   [-aocvm_guardband] [-pocvm_guardband] [-pocvm_coefficient_scale_factor]
   [-increment]
   [-significant_digits digits]
   [-nosplit]
   [object_list]
```

Data Types

digits	int
object_list	list

ARGUMENTS

-include_inherited

Reports the name of the design, ancestor cell, or library cell from which each object inherits its derating settings.

-variation

Reports derating previously set by the **set_timing_derate -variation** ... command. This option should be used only with the PrimeTime VX tool.

-aocvm_guardband

Reports derating previously set by the **set_timing_derate -aocvm_guardband** ... command.

-pocvm_guardband

Reports derating previously set by the **set_timing_derate -pocvm_guardband** ... command.

-pocvm_coefficient_scale_factor

Reports derating previously set by the **set_timing_derate -pocvm_coefficient_scale_factor** ... command.

-increment

Reports incremental derating adjustments previously set by the **set_timing_derate -increment** ... command.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for values in the report. The default is determined by the **report_default_significant_digits** variable.

-nosplit

Prevents splitting of long lines of text in the report.

object_list

Reports derating for the given list or collection of cells, library cells, or nets. To report the names of parent objects from which these objects inherit their derating factors, also use the **-include_inherited** option in the command.

DESCRIPTION

The **report_timing_derate** command reports derating previously set by the **set_timing_derate** command for each object on which derating has been set: the design, cell instances, and nets; or for the objects listed in the *object_list* option. The derating factors are reported in a table format, organized by setting type:

```
pt_shell> report_timing_derate -significant_digits 2
 ...
                          ----- Clock -----             ----- Data -----
                         Rise          Fall            Rise          Fall
                      Early  Late    Early  Late    Early  Late    Early  Late
-------------------------------------------------------------------------------
design: MYDESIGN
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Net delay dynamic   0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Cell delay          0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Cell check           --     --      --     --      --     --      --     --

cell (hier): DESIGN_TOP/I_RISC_CORE/I_ALU
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Net delay dynamic   0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Cell delay          0.85   1.20    0.85   1.10    0.85   1.20    0.85   1.10
  Cell check           --     --      --     --      --     --      --     --

net: net198
  Net delay static    0.90   1.30    0.90   1.30    0.90   1.30    0.90   1.30
  Net delay dynamic   0.90   1.30    0.90   1.30    0.90   1.30    0.90   1.30
```

To report the name of the design, ancestor cell, or library cell from which each object inherits its derating settings, use the **-include_inherited** option in the command. For example, a leaf-level cell might show the name of the ancestor cell or library cell on which derating was set.

EXAMPLES

In the following example, derating is set globally (only on the design), so the **report_timing_derate** command reports only global derating factors.

```
pt_shell> set_timing_derate  -data -early 0.92
pt_shell> set_timing_derate  -data  -late  1.14
```

```
pt_shell> set_timing_derate -clock -early 0.75
pt_shell> set_timing_derate -clock  -late  1.26
pt_shell> report_timing_derate
 ...
                            ----- Clock ------              ------ Data ------
                           Rise            Fall            Rise            Fall
                       Early    Late   Early    Late   Early    Late   Early    Late
 ---------------------------------------------------------------------------------
 design: simple_path
   Net delay static     0.75    1.26    0.75    1.26    0.92    1.14    0.92    1.14
   Net delay dynamic    0.75    1.26    0.75    1.26    0.92    1.14    0.92    1.14
   Cell delay           0.75    1.26    0.75    1.26    0.92    1.14    0.92    1.14
   Cell check            --      --      --      --      --      --      --      --
```

In the following example, incremental derating is set globally (only on the design), so the **report_timing_derate -increment** command reports only global incremental derating adjustments.

```
pt_shell> set_timing_derate  -data -increment -early -0.02
pt_shell> set_timing_derate  -data -increment  -late  0.06
pt_shell> set_timing_derate -clock -increment -early -0.05
pt_shell> set_timing_derate -clock -increment  -late  0.10
pt_shell> report_timing_derate -increment
 ...
                            ----- Clock ------               ------ Data ------
                           Rise            Fall             Rise            Fall
                       Early    Late   Early    Late    Early    Late   Early    Late
 ----------------------------------------------------------------------------------
 design: simple_path
   Net delay static    -0.05    0.10   -0.05    0.10    -0.02    0.06   -0.02    0.06
   Net delay dynamic   -0.05    0.10   -0.05    0.10    -0.02    0.06   -0.02    0.06
   Cell delay          -0.05    0.10   -0.05    0.10    -0.02    0.06   -0.02    0.06
   Cell check            --      --      --      --       --      --      --      --
```

In the following example, the commands set derating globally on the design TOP and more narrowly on the hierarchical block H1. The **report_timing_derate** command reports the derating factors that apply to the lower-level cell H1/U5.

```
pt_shell> set_timing_derate -early 0.95
pt_shell> set_timing_derate -late  1.06
pt_shell> set_timing_derate -early 0.82 [get_cells H1]
pt_shell> set_timing_derate -late  1.12 [get_cells H1]
pt_shell> report_timing_derate -include_inherited [get_cells H1/U5]
 ...

                            ----- Clock ------              ------ Data ------
                           Rise            Fall            Rise            Fall
                       Early    Late   Early    Late   Early    Late   Early    Late
 ---------------------------------------------------------------------------------
 cell (hier): H1/U5
   Net delay static    0.95    1.06    0.95    1.06    0.95    1.06    0.95    1.06
   Inherited            TOP     TOP     TOP     TOP     TOP     TOP     TOP     TOP

   Net delay dynamic   0.95    1.06    0.95    1.06    0.95    1.06    0.95    1.06
   Inherited            TOP     TOP     TOP     TOP     TOP     TOP     TOP     TOP

   Cell delay          0.82    1.12    0.82    1.12    0.82    1.12    0.82    1.12
   Inherited             H1      H1      H1      H1      H1      H1      H1      H1

   Cell check           --      --      --      --      --      --      --      --
   Inherited            --      --      --      --      --      --      --      --
```

SEE ALSO

```
set_timing_derate(2)
reset_timing_derate(2)
```

report_transitive_fanin

Reports logic in fanin of specified objects.

SYNTAX

```
string report_transitive_fanin
   [-nosplit]
   [-from from_list]
   [-through through_list]
   -to to_list
   [-trace_arcs timing | enabled | all]
```

Data Types

```
from_list          list
through_list       list
to_list            list
```

ARGUMENTS

-nosplit

Does not split lines if columns overflow.

-from *from_list*

Specifies a list of pins, ports, and nets in the design to constrain the fanin of *to_list* that is reported. If you specify a net, the effect is same as listing all the load pins on the net.

-through *through_list*

Specifies a list of pins, ports, and nets in the design to constrain the fanin of *to_list* that is reported. If you specify a net, the effect is same as listing all the pins on the net.

-to *to_list*

Specifies a list of pins, ports, or nets in the design, whose transitive fanin is reported. If you specify a net, the effect is the same as listing all driver pins on the net.

-trace_arcs timing | enabled | all

Specifies the type of combinational arcs to trace during the traversal. Allowed values are

- **timing** (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- **enabled** - Permits the tracing of all enabled arcs and disregards case analysis values.

- **all** - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

DESCRIPTION

The command produces a report showing the transitive fanin of specified pins, ports, or nets in the design. A pin is considered to be in the transitive fanin of an object if there is a timing path through combinational logic from the pin to that object (also see the **-trace_arcs** option). The fanin report stops at the clock pins of registers (sequential cells).

If the **current_instance** command is set, the report focuses on the fanin within the **current_instance** command. The report stops at the boundaries of the **current_instance** command, and no paths outside the **current_instance** command are reported. The report shows the hierarchical boundary pins on the current instance.

EXAMPLES

The following example shows the transitive fanin of a pin in the design.

```
pt_shell> report_transitive_fanin -to FF1/D

****************************************
Report : transitive_fanin
Design : top
 ...
****************************************

Fanin network of sink 'FF1/D':

Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
gate1/Y          FF1/D             (net arc)   same


Load Pin         Driver Pin        Type        Sense
-------------------------------------------------------------
gate1/A          gate1/Y           AN2         same
gate1/B          gate1/Y           AN2         same


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
IN1              gate1/A           (net arc)   same
IN2              gate1/B           (net arc)   same
```

SEE ALSO

```
current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
```

```
report_transitive_fanout(2)
```

report_transitive_fanout

Reports logic in fanout of specified objects.

SYNTAX

```
string report_transitive_fanout
   [-nosplit]
   [-trace_arcs timing | enabled | all]
   -clock_tree
   -from from_list
   [-through through_list]
   [-to to_list]
```

Data Types

```
from_list          list
through_list         list
to_list            list
```

ARGUMENTS

-nosplit

Does not split lines if columns overflow.

-trace_arcs timing | enabled | all

Specifies the type of combinational arcs to trace during the traversal. Allowed values are

- **timing** (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- **enabled** - Permits the tracing of all enabled arcs and disregards case analysis values.
- **all** - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

-clock_tree

Reports transitive fanout of all clock sources in the design. When this option is used, the fanout search is constrained to objects in the clock network only.

-from *from_list*

Specifies a list of pins, ports, and nets in the design whose transitive fanout is reported. If you specify a net, the effect is same as listing all the load pins on the net.

-through *through_list*

Specifies a list of pins, ports, and nets in the design to constrain the fanout of *from_list* that is reported. If you specify a net, the effect is same as listing all the pins on the net.

-to ***to_list***

Specifies a list of pins, ports, and nets in the design to constrain the fanout of *from_list* that is reported. If you specify a net, the effect is same as listing all the driver pins on the net.

DESCRIPTION

Produces a report showing the transitive fanout of specified pins or ports in the design. A pin is considered to be in the transitive fanout of an object if there is a timing path through combinational logic from the object to that pin (also see the **-trace_arcs** option). The fanout report stops at the inputs to registers (sequential cells).

If the **current_instance** command is set, the report focuses on the fanout within the **current_instance** command. The report stops at the boundaries of the **current_instance** command, and no paths outside the **current_instance** command are reported. The report also shows the hierarchical boundary pins on the **current_instance** command.

EXAMPLES

The following example shows the transitive fanout of a pin in the design.

```
pt_shell> report_transitive_fanout -from FF1/Q

****************************************
Report : transitive_fanout
Design : top
 ...
****************************************

Fanout network of source 'FF1/Q':


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
FF1/Q            gate5/A           (net arc)   same


Load Pin         Driver Pin        Type        Sense
-------------------------------------------------------------
gate5/A          gate5/Y           AN2         same


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
gate5/Y          OUT2              (net arc)   same
```

The following command shows the transitive fanout of all the clock sources in the design, constrained to the clock network.

```
pt_shell> report_transitive_fanout -clock_tree

****************************************
```

```
Report : transitive_fanout
        -clock_tree
Design : top
 ...
****************************************

Fanout network of source 'CLK':

Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
CLK              FF3/CLK           (net arc)   same
CLK              FF2/CLK           (net arc)   same
CLK              FF1/CLK           (net arc)   same
```

SEE ALSO

```
create_clock(2)
current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
report_transitive_fanin(2)
```

report_units

Reports the unit information.

SYNTAX

```
string report_units
   [-nosplit]
```

ARGUMENTS

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output..SH "DESCRIPTION" The **report_units** command displays the units used by the **current_design**. To generate the report, first load and link the design to a library. The units are always reported in reference to the MKS units, like Farad, Amp, Ohm, Second and Volt. The resistance unit is inferred from the time and capacitance unit. Both time and capacitance units are required for PrimeTime. If either of them is missing, the unit is undefined and it is reported as "Not specified." The most common cause of this problem is the library is in an old format and it needs to be recompiled with the latest version of Library Compiler.

If power analysis is on, the power units will also be reported.

EXAMPLES

The following example shows how to report the unit information.

```
pr_shell> report_units

****************************************
Report : units
Design : simple
Version: X-2005.06-Beta2
Date   : Mon Apr 28 11:55:27 2005
****************************************

Units
---------------------------------------------
Capacitive_load_unit        : 1e-12 Farad
Current_unit                : 0.001 Amp
Inductance_unit             : 1e-09 Henry
Resistance_unit             : 1000 Ohm
Time_unit                   : 1e-09 Second
```

```
  Voltage_unit                : 1 Volt
```

SEE ALSO

```
read_db(2)
set_min_library(2)
```

report_vcd_hierarchy

Reports the hierarchy in the VCD file.

SYNTAX

```
status report_vcd_hierarchy
   [-pipe_exec shell_command_string]
   [-full_path]
   [-find pattern_string]
   [file_name]
```

Data Types

```
shell_command_string     string
pattern_string           string
file_name                string
```

ARGUMENTS

-pipe_exec *shell_command_string*

Specifies a shell command which is used to generate the VCD file *file_name*. This option invokes the command and directly pipe the output VCD file to PrimeTime-PX. In another word, the simulation and power analysis are in parallel run. No VCD disk file is generated at all.

-full_path

Displays each scopes in full path format.

-find *pattern_string*

Report only those scopes whose last hierarchy name is matched the specified *pattern*. They are always reported in full path format. Nothing is printed out if not found.

file_name

Specifies a file in VCD format from which the switching activity information is to be read. If file_name ends with .gz, .Z, .vpd and .fsdb, it is read as a gzipped file, a compressed file, a VCD+ file, and a FSDB file, respectively, otherwise it is just read as a normal ASCII VCD file. If the *file_name* is omitted, the name is taken from the **read_vcd** command if it exists; otherwise, an error message is issued.

DESCRIPTION

This command displays the indented hierarchy in the specified VCD file.

To read a gzip VCD file, a compressed VCD file, VCD+ file, or FSDB file, your PATH needs to include gunzip, uncompress, vpd2vcd, and fsdb2vcd. vpd2vcd and fsdb2vcd come with Synopsys VCS and Novas Debussy, respectively.

The pipe option allows PrimeTime PX reading in VCD file directly from simulator, which avoids generating huge VCD file. The VCD file name in this scenario is just served as a pipe name. After running the VCD file does not exist. You don't need to change your existing test bench. Use it just like normal VCD dump (inserting $dumpfile, $dumpvars, etc.) You do not need to invoke the simulator, either. PrimeTime PX automatically runs the simulation and directly read in the VCD output from the simulator.

EXAMPLES

The following example shows a VCD hierarchy report.

```
pt_shell> report_vcd_hierarchy vcd.dump.gz
tb
  macinst
    U3
    U4
    U7
    U9
    U10
    U11
    U12
```

SEE ALSO

read_vcd(2)

report_virtual_decap_cells

Report virtual decap cells inserted to the design.

SYNTAX

```
status report_virtual_decap_cells
   -connect_to_power  power_net_name
   -connect_to_ground ground_net_name
   [-verbose]
```

Data Types

```
string power_net_name
string ground_net_name
```

ARGUMENTS

-connect_to_power *power_net_name*

Specifies the power supply net to which to connect power pins of the virtual decap cells.

-connect_to_ground *ground_net_name*

Specifies the ground supply net to which to connect ground pins of the virtual decap cells.

-verbose

Specifies to generate a detailed report.

DESCRIPTION

The **report_virtual_decap_cells** command generates a report of the virtual decap cells inserted to the design. Only the virtual decap cells whose power and ground pins are connected to the specified power and ground supply nets are included in the report.

EXAMPLE

The following example reports the virtual decap cells inserted between the power supply net VDD and ground supply net VSS by **insert_virtual_decap_cells** command(s).

```
pr_shell> report_virtual_decap_cells -connect_to_power VDD -connect_to_ground VSS
****************************************
Report : Virtual decap cells inserted
```

```
Design : design
Version: J-2014.06-alpha
Date   : Mon Mar 24 15:07:45 2014
****************************************

Coupled power domains: VDD VSS

Total number of virtual decap cells in the coupled power domains: 13
Total decoupling capacitance inserted to the coupled power domains:      1.000 [pF]
1
```

The following example reports the virtual decap cells inserted between the power supply net VDD and ground supply net VSS by **insert_virtual_decap_cells** command(s). A detailed report is generated due to the -verbose option.

```
pr_shell> report_virtual_decap_cells -connect_to_power VDD -connect_to_ground VSS -verbose
****************************************
Report : Virtual decap cells inserted
Design : design
Version: J-2014.06-alpha
Date   : Mon Mar 24 15:07:45 2014
****************************************

Coupled power domains: VDD VSS

Format: instance_name virtual_decap_value [pF]
INV11      0.077 [pF]
ARCR0_1    0.077 [pF]
INV21      0.077 [pF]
INV31      0.077 [pF]
INV41      0.077 [pF]
FEED0_14   0.077 [pF]
INV51      0.077 [pF]
FEED0_5    0.077 [pF]
INV61      0.077 [pF]
FEED0_4    0.077 [pF]
FEED0_3    0.077 [pF]
FEED0_2    0.077 [pF]
FEED0_1    0.077 [pF]

Total number of virtual decap cells in the coupled power domains: 13
Total decoupling capacitance inserted to the coupled power domains:      1.000 [pF]
1
```

SEE ALSO

insert_virtual_decap_cells(2)
remove_virtual_decap_cells(2)
calculate_rail_voltage(2)

report_voltage_regulator_lib_cell

Reports library information for the voltage regulator library cells created by the **set_voltage_regulator_lib_cell** command.

SYNTAX

```
status report_voltage_regulator_lib_cell
   [cell_master_list]
```

Data Types

```
list cell_master_list
```

ARGUMENTS

cell_master_list

Specifies the names of the voltage regulator cell masters whose related library information is to be reported. Items within the list can only be cell master names. If no list is given, the command reports all cell masters that are created by the **set_voltage_regulator_lib_cell** command.

DESCRIPTION

The command reports library information on the voltage regulator cell masters that are created with the **set_voltage_regulator_lib_cell** command. Run the command before the **open_mw_cel** command; otherwise the command run will fail.

By default, the command reports input power pin names, output power pin names, ground pin names, reference pin names, other input/output pin names and associated supply voltages.

EXAMPLES

The following example creates a voltage regulator library cell for the cell master VR1, then runs the **report_voltage_regulator_lib_cell** command to report library information for the cell master VR1.

```
pr_shell> set_voltage_regulator_lib_cell VR1 \
            -input_power_pin {VCC 3.0} -output_power_pin {VDD 1.3} \
            -reference_pin {REF 1.3} -ground_pin {VSS 0.0} \
            -other_io_pin {B 3.0}
```

```
pr_shell> report_voltage_regulator_lib_cell
****************************************
Report : voltage regulator library cells
Design :
****************************************

  Lib cell       Input power    Output power   Ground         Reference      Other io
                 pin            pin            pin            pin            pin
--------------------------------------------------------------------------------
  VR1            VCC (3V)       VDD (1.3V)     VSS (0V)       REF (1.3V)     B (3.0V)
1
```

SEE ALSO

set_voltage_regulator_lib_cell(2)

report_wire_load

Reports wire load information.

SYNTAX

```
string report_wire_load
   [-nosplit]
   [cell_list]
```

Data Types

```
cell_list     list
```

ARGUMENTS

-nosplit

Does not split lines if the column overflows.

cell_list

Specifies a list of hierarchical cells.

DESCRIPTION

The **report_wire_load** command shows the characteristics of wire load models set on the current design and on cells of the current design. If you do not specify a cell list, by default the command displays the wire load models and wire load mode used in the current design. If you specify a cell list, the command displays the wire load models used in the specified cells.

EXAMPLES

This command displays the wire load model in the current design.

```
pt_shell> report_wire_load

****************************************
Report : wire_load
Design : top
```

```
 ...
****************************************

Wire Loading Model Mode: segmented

Cell          Design       Wire_model  Library       Selection_type
-----------------------------------------------------------------------
              inv_top      WLM1        my_lib        user-specified
cell          tst          WLM2        my_lib        user-specified

1
```

SEE ALSO

```
remove_wire_load_model(2)
remove_wire_load_selection_group(2)
report_design(2)
set_wire_load_min_block_size(2)
set_wire_load_model(2)
set_wire_load_selection_group(2)
```

reset_mode

Resets cell mode groups or design mode groups to the default state.

SYNTAX

```
status reset_mode
   [-type cell | design]
   [-group group_list]
   [instance_list]
```

Data Types

```
group_list          list
instance_list       list
```

ARGUMENTS

-type cell | design

Indicates the type of mode group to be set to default. This option has the following mutually-exclusive valid values: *design* and *cell*.

- The *cell* value specifies that cell mode groups are to be set to default. Cell mode groups are defined on library cells in the library.
- The *design* value specifies that design mode groups are to be set to default.

  Design mode groups are user specified using the **define_design_mode_group** command. If the *-type* option is omitted from the command, this is equivalent to specifying *-type cell*.

-group *group_list*

Specifies a list of mode groups, each of which is to be reset to its default setting. If the *-type* option has a cell value, the *group_list* option must contain only cell mode groups. If the *-type* option has a design value, the *group_list* option must contain only design mode groups.

instance_list

Specifies a list of instances for which the specified cell mode groups are to be set to default. If no instance list is specified, all moded cells are considered. This list must only be included with the *-type cell* option. No error occurs if you reset the modes on a cell that has no modes.

DESCRIPTION

Resets mode groups to the default state of all modes enabled. Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell.

When a cell mode group is set to default for a given instance of the library cell all of its modes are enabled for that cell. All timing arcs of the enabled modes also get enabled with respect to modes for that cell.

Cell mode groups can have different states due to the **set_mode -type cell**, **set_mode -type design**, and conditional evaluation. When conflict arises the following precedence rule is employed:

- **set_mode -type cell**
- **set_mode -type_design**
- Evaluation of mode conditions

To reset the state of the cell mode group due to the **set_mode -type cell** command, use the **reset_mode -type cell** command.

To reset the state of the cell mode group due to the **set_mode -type design** command, use either the **reset_mode -type design** or **remove_design_mode** command.

To reset the state of the cell mode group due to conditional evaluation, use the **remove_case_analysis** command or edit the Verilog netlist to remove logical constants.

You define design modes and design mode groups using the **define_design_group** and **map_design_mode** commands. Design modes can be mapped to a set of cell modes, set of paths, or both cell modes and paths. When a design mode group is reset to default all design modes of that group are enabled. When a design mode is enabled all paths mapped to the design mode are reset and all previous setting of cell modes mapped to the design modes are removed.

EXAMPLES

In the following example, we have two cell mode groups RW and SH defined on cell Uram1/D0. The first command sets READ as the active mode for RW and EARLY as the active mode for SH.

```
pt_shell> set_mode {EARLY READ} Uram1/D0
```

The following command can be employed to reset the cell mode groups to default.

```
pt_shell> reset_mode -group {RW SH} Uram1/D0
```

To reset all cell mode group in Uram1/DO use the following command

```
pt_shell> reset_mode Uram1/D0
```

To reset all cell mode groups in the design use

```
pt_shell> reset_mode
```

In the following example, the first command defines two design modes, DM1 and DM2. (Since no mode group name was specified, the mode group is named "default".) The second command specifies that for design mode DM1, the READ cell mode is active for the RAM instance Uram1. The third command specifies that for design mode DM2, the WRITE cell mode is active for the RAM instance Uram1. The fourth command maps all paths ending at Uram1/D0 to

the design mode DM2. The fifth command sets the design_mode DM1. The sixth command sets READ as the active cell mode and disables the WRITE cell mode.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode DM1 READ Uram1
pt_shell> map_design_mode DM2 WRITE Uram1
pt_shell> map_design_mode DM2 -to Uram1/D0
pt_shell> set_mode -type design DM1
pt_shell> set_mode -type cell READ Uram1
```

The following example resets the design mode group "default".

```
pt_shell> reset_mode -type design -group default
```

This command enables the design_mode DM2 resulting in the resetting of all paths mapped to Uram1/D0. However the cell mode WRITE on Uram1/D0 does not get enabled since it was also disabled by command 5 above. To reset the cell mode group to default use

```
pt_shell> reset_mode -type cell Uram1
```

The following command is equivalent to the previous command

```
pt_shell> reset_mode Uram1
```

SEE ALSO

define_design_mode_group(2)
remove_design_mode(2)
report_mode(2)
map_design_mode(2)
set_mode(2)
set_case_analysis(2)

reset_noise_parameters

Resets the noise analysis parameters for the current design.

SYNTAX

```
int reset_noise_parameters
```

ARGUMENTS

None.

DESCRIPTION

The **reset_noise_parameters** command resets the parameters that are considered during the noise analysis. If this command is used, the default settings are used for the noise analysis: beyond the rail analysis is ignored, arrival times of aggressors are considered and propagation of noise is disabled.

EXAMPLES

The following example resets the default noise parameters:

```
pt_shell> reset_noise_parameters
```

SEE ALSO

```
report_noise(2)
report_noise_parameters(2)
set_noise_parameters(2)
update_noise(2)
si_noise_effort_threshold_within_rails(3)
```

reset_path

Resets specified paths to single-cycle behavior.

SYNTAX

```
status reset_path
   [-setup | -hold]
   [-rise | -fall]
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
```

Data Types

```
from_list               list
rise_from_list          list
fall_from_list          list
through_list            list
rise_through_list       list
fall_through_list       list
to_list                 list
rise_to_list            list
fall_to_list            list
```

ARGUMENTS

-setup

Indicates that only setup (maximum delay) evaluation is reset to its default, single-cycle behavior. If neither the **-setup** nor **-hold** options is specified, both setup and hold checking are reset to single-cycle.

-hold

Indicates that only hold (minimum delay) evaluation is reset to its default, single-cycle behavior. If neither the **-setup** nor **-hold** options is specified, both setup and hold checking are reset to single-cycle.

-rise

Indicates that only rising path delays are reset to single-cycle behavior. If neither the **-rise** nor **-fall** options is specified, both rising and falling delays are reset to single-cycle.

-fall

Indicates that only falling path delays are reset to single-cycle behavior. If neither the **-rise** nor **-fall** options is specified, both rising and falling delays are reset to single-cycle.

-from ***from_list***

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the following options: **-from**, **-rise_from**, or **-fall_from**.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: **-from**, **-rise_from**, or **-fall_from**.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: **-from**, **-rise_from**, or **-fall_from**.

-through ***through_list***

Specifies a list of pins, ports, and nets through which the paths must pass that are reset. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit the **-through** option, all timing paths specified using the **-from** and **-to** options are affected.

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one

path endpoint on that cell is affected. You can use only one of the following options: **-to**, **-rise_to**, or **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: **-to**, **-rise_to**, or **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: **-to**, **-rise_to**, or **-fall_to**.

DESCRIPTION

The command restores designated timing paths in the current design to the default single-cycle behavior. Use the **reset_path** command to undo the effect of the **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay** commands. Attributes set by these commands are removed by the **reset_path**. command.

Note that a general **reset_path** command removes the effect of matching general or specific point-to-point exception commands, if there is a matching object in the path specification of the original exception-setting commands. For example, the following command

```
reset_path -from {CLK}
```

resets more specific matching exceptions for the object CLK, such as

```
set_false_path -from {CLK} -to {d/Z}
set_false_path -from {CLK} -to {g/Z}
```

If you do not specify either the **-setup** or **-hold** option, the default behavior is restored to both. The default is a setup relation of one cycle and a hold relation of zero cycles, so that hold is checked one active edge before the setup data at the destination register.

Setup and hold information is different for rising and falling transitions. In most cases, these are reset together. Use the **-rise** or **-fall** options to reset only one or the other. To disable setup or hold calculations for paths, use the **set_false_path** command.

The general and specific constraint options (for example, **-through** and **-rise_through** or **-fall_through**) are treated as different qualifiers. That is, if you issue the **reset_path -fall_through** variable, only those constraints are removed that were set with the **-fall_through** option; any that were set with the **-through** option are not removed. The same applies to the general and specific **-to** and **-from** options. For an example, see the EXAMPLES section.

EXAMPLES

The following command resets the timing relation between ff1/CP and ff2/D to the default single-cycle behavior.

```
pt_shell> reset_path -from { ff1/CP } -to { ff2/D }
```

The following command resets only the rising delay to single cycle for all paths ending at latch2d.

```
pt_shell> reset_path -rise -to latch2d
```

The following example first sets a specific and a general point-to-point exception, then removes the exceptions.

```
pt_shell> set_multicycle_path 2 -from A -to Z
pt_shell> set_max_delay 15 -to Z
pt_shell> reset_path -to Z
```

In following example, the **reset_path** command removes only the max delay constraints, because the **-to** and **-rise_to** options are treated as different qualifiers and do not match.

```
pt_shell> set_multicycle_path 2 -from A -to Z
pt_shell> set_max_delay 15 -rise_to Z
pt_shell> reset_path -rise_to Z
```

SEE ALSO

current_design(2)
report_constraint(2)
reset_design(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

reset_power_derate

Resets power derating factors set either on a design or on a specified list of instances (cells or library cells).

SYNTAX

```
int reset_power_derate
   [-groups group_name_list]
   [-rails rail_list]
   object_list
```

Data Types

```
group_name_list     list
rail_list           list
object_list         list
```

ARGUMENTS

-groups *group_name_list*

Specifies the power group or groups in which the derate factors of all the cells are reset.

-rails *rail_list*

Specifies a list of power supply nets in which the derate factors of all the objects are reset.

object_list

Specifies current design or a list of cells or library cells on which the power derating factors are reset to 1.0.

DESCRIPTION

Invoke this command with no arguments to reset all derate factors set on the design (to the default of 1.0). This includes both derating factors set globally on the design and those set on specific instances in the design. If the command is called with a list of objects, only the power derating factors on the specified objects are reset. After a **reset_power_derate** command is applied to an object, the power derating factors of all power components, including switching, internal, and leakage power are reset. If the *object_list* variable contains the current design, then only global power derating factors are reset and instance specific power derating factors remain untouched.

If the *object_list* variable contains any hierarchical cells, then all cells within that hierarchical cell and also all cells of lower hierarchies will have their inherited power derating factors updated accordingly. Note that power derating factors set specifically on an object are not overwritten by a set or reset command on its parent hierarchical cell. To reset the power derating factors set specifically on the child cells, include them in the object list of the

command.

The **-groups** options allows to reset the derate factors of cells only in the specified groups.

Heterogeneous collections of objects can be combined and passed to this command.

If the *object_list* variable contains any hierarchical cells, then all cells within that hierarchical cell and also cells of all lower hierarchies will have their inherited derate factors updated accordingly. Note that derate factors set specifically on an object are not overwritten by a set or reset command on its parent hierarchical cell.

EXAMPLES

The following command resets both power derating factors set globally on the design and those set on specific instances in the design.

```
pt_shell> reset_power_derate
```

The following command resets the power derating factors set globally on the design MY_DESIGN only.

```
pt_shell> reset_power_derate [get_design MY_DESIGN]
```

The following example resets the power derating factors set on cell U1.

```
pt_shell> reset_power_derate [get_cell U1]
```

The following command resets the power derating factors set on all cells matching "*".

```
pt_shell> reset_power_derate [add_to_collection [get_cells *]
```

The following example resets the power derating factors set on all instances of library cell IV in the library MY_LIB.

```
pt_shell> reset_power_derate [get_lib_cells MY_LIB/IV]
```

Assuming that the hierarchical cell H1 contains a cell U1, then the following command resets the derate factors set on U1. As a result, U1 inherits the derate factors that are set on H1.

```
pt_shell> reset_power_derate [get_cells H1/U1]
```

Assuming that the hierarchical cell H1 also contains a hierarchical cell H2, then the following command resets the power derate factors set on H2. As a result, H2 and all of its cells inherit the power derating factors that are set on H1.

```
pt_shell> reset_power_derate [get_cells H1/H2]
```

SEE ALSO

```
report_power(2)
report_power_derate(2)
set_power_derate(2)
```

reset_rtl_to_gate_name

Resets all the RTL-to-gate name mapping information.

SYNTAX

```
integer reset_rtl_to_gate_name
```

ARGUMENTS

None.

DESCRIPTION

The **reset_rtl_to_gate_name** command resets all the RTL-to-gate name mapping information stored previously using the **set_rtl_to_gate_name** and **unset_rtl_to_gate_name** commands.

EXAMPLES

The following example provides the resetting of the RTL-to-gate name mapping.

```
pt_shell> reset_rtl_to_gate_name
```

SEE ALSO

set_rtl_to_gate_name(2)
unset_rtl_to_gate_name(2)

reset_scale_parasitics

Resets the parasitic scaling factors previously set by the **scale_parasitics** command.

SYNTAX

```
status reset_scale_parasitics
   [net_list]
```

Data Types

```
net_list       list
```

ARGUMENTS

net_list

Limits the resetting of parasitic scaling factors to only the listed nets. By default, the command resets all net-specific and global parasitic scaling settings. Using this option, only nets with net-specific scaling factors can be reset.

DESCRIPTION

The **reset_scale_parasitics** command resets the parasitic resistance and capacitance scaling factors previously set by the **scale_parasitics** command. By default, both the global and net-specific scaling factor settings are reset.

You can optionally restrict the resetting to only a specified list of nets that have net-specific scaling factor settings.

EXAMPLES

The following example resets all parasitic scaling factors.

```
pt_shell> reset_scale_parasitics
1
```

SEE ALSO

```
read_parasitics(2)
report_scale_parasitics(2)
scale_parasitics(2)
```

reset_switching_activity

Resets switching activity annotation on nets, pins, ports, and cells of the current design.

SYNTAX

```
int reset_switching_activity
   [-static_probability]
   [-toggle_rate]
   [-state_condition state]
   [-path_sources name_list]
   [-no_hierarchy]
   [object_list]
   [-quiet]
```

Data Types

```
state           string
name_list       string
object_list     list
```

ARGUMENTS

-static_probability

Specifies the static probability value to reset for the given design object. The static probability value is internally reset to an invalid number.

-toggle_rate

Specifies the toggle rate (rise and fall, if applicable) and glitch rate (rise and fall, if applicable) to reset for the given design object. The toggle and glitch rate values are internally reset to invalid numbers.

-state_condition *state*

Specifies the state condition when resetting state-dependent toggle rate and glitch rates on pins or state-dependent static probabilities on cells. State-dependent toggle rate and glitch rate can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal/leakage power characterization. The state condition should be enclosed between " ". Moreover, if you want to reset switching activity information for the default state condition, the argument of the *-state* option should be "default". Note that the *-state_condition* option applies only to the leaf level cells.

-path_sources *name_list*

Specifies the path sources when resetting path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path sources specified with this argument must be

the same as those in the internal power characterization. When listing more than one pin in path sources, the pin names should be separated by a space and enclosed between quotes " ". For example, if the pins in path sources are A and B, the argument for the *-path* option looks like "A B". The *-path_sources* argument applies only to the leaf level cells. Note that if the *name_list* option is given using the *-path_sources* argument, the *state* option should also be provided using the *-state_condition* option.

-no_hierarchy

This option specifies that only the objects in the hierarchy of the list of instances, specified using the *object_list* option, is reset. If no instance list is specified, then the switching activity on the objects in the same hierarchy of the current instance is reset. If the *-no_hierarchy* option is not specified and the *object_list* option is also not given, the switching activity on all the objects in all the hierarchy of the current instance are reset. Note that if leaf level cells are given using the *object_list* option, the command internally assumes this option is true.

object_list

Specifies a list of nets, pins, ports, or instances in the current design on which the static probability, toggle rate, and glitch rate switching activity values are reset.

-quiet

Specifies quiet mode and suppresses warnings on design objects for which switching activity could not be reset.

DESCRIPTION

Use this command to reset design nets, ports, pins, and cells with the different kinds of switching activity. These include simple toggle rate, glitch rate, and static probability on nets, ports and pins; state-dependent and path-dependent toggle, and glitch rate on cell pins, and state-dependent static probabilities on cells. Toggle and glitch rates, and static probabilities are reset using the *-toggle_rate* and *-static_probability* options, respectively. You can make the toggle rate, glitch rate, and static probability state-dependent by specifying the *state* option with the *-state_condition* option. You can make the toggle rate and glitch rate path-dependent by specifying the path sources with the *-path_sources* option. The design objects that are reset with switching activity can be specified as a list of objects. You can reset only the objects in the same hierarchy by using the *-no_hierarchy* option.

Most of the above functionality and all the options apply when the *power_analysis_mode* variable is set to the *averaged* or *leakage_variation* option . When the*power_analysis_mode* variable is set to the *time_based* option, the **reset_switching_activity** command has a different meaning. In this mode, the **reset_switching_activity** command removes any VCD or other activity indicated previously with the **read_vcd** command. Because the *time_based* mode is not based on toggle rates, attempting to reset switching activity on individual objects has no effect.

EXAMPLES

The following example shows resetting toggle rate, glitch rate, and static probability values only on the current instance of the TOP_MULT design:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity -no_hierarchy

pt_shell> reset_switching_activity -no_hierarchy -toggle_rate -static_probability
```

```
pt_shell> reset_switching_activity -toggle_rate -no_hierarchy

pt_shell> reset_switching_activity -static_probability -no_hierarchy
```

The following example shows resetting the toggle rate, glitch rate, and static probability values on the whole TOP_MULT design:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity

pt_shell> reset_switching_activity -toggle_rate -static_probability

pt_shell> reset_switching_activity -toggle_rate

pt_shell> reset_switching_activity -static_probability
```

The following example shows resetting the toggle rate, glitch rate, and static probability values on the list of instances:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity {U1 U2}

pt_shell> reset_switching_activity -no_hierarchy {U1 U2}

pt_shell> reset_switching_activity -toggle_rate {U1 U2}

pt_shell> reset_switching_activity -static_probability {U1 U2}

pt_shell> reset_switching_activity -no_hierarchy -toggle_rate {U1 U2}

pt_shell> reset_switching_activity -no_hierarchy -static_probability
         {U1 U2}
```

The following **reset_switching_activity** command resets simple toggle rate value, glitch rate value, and static probability value on all design input ports:

```
pt_shell> reset_switching_activity -toggle_rate
        -static_probability [get_inputs]

pt_shell> reset_switching_activity [get_inputs]
```

The following example resets different switching activity values on design output ports:

```
pt_shell> reset_switching_activity -toggle_rate [get_outputs]

pt_shell> reset_switching_activity -static_probability
         [get_outputs]

pt_shell> reset_switching_activity -static_probability
        -toggle_rate [get_outputs]
```

The following example resets state-dependent static probabilities on the or1 cell:

```
pt_shell> reset_switching_activity -static_probability
        -state_condition "A & B" [get_cell or1]

pt_shell> reset_switching_activity -state_condition "A & B"
        [get_cell or1]

pt_shell> reset_switching_activity -static_probability
        -state_condition "A & ! B" [get_cell or1]
```

```
pt_shell> reset_switching_activity -static_probability
        -state_condition "! A & B" [get_cell or1]

pt_shell> reset_switching_activity -static_probability
        -state_condition "! A & ! B" [get_cell or1]

pt_shell> reset_switching_activity -static_probability
        -state_condition "default" [get_cell or1]
```

The following example resets simple and path sources, dependent toggle rates, and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> reset_switching_activity -toggle_rate [get_pin xor1/Y]

pt_shell> reset_switching_activity -toggle_rate -path_sources "A"
        -state_condition "B" [get_pin xor1/Y]

pt_shell> reset_switching_activity -path_sources "A"
        -state_condition "B" [get_pin xor1/Y]

pt_shell> reset_switching_activity -toggle_rate -path_sources "B"
        -state_condition "A" [get_pin xor1/Y]

pt_shell> reset_switching_activity -toggle_rate -path_sources "A B"
        -state_condition "default" [get_pin xor1/Y]
```

SEE ALSO

get_switching_activity(2)
read_saif(2)
report_power(2)
report_switching_activity(2)
set_switching_activity(2)

reset_timing_derate

Resets the derating previously set by the **set_timing_derate** command.

SYNTAX

```
status reset_timing_derate
   [-hierarchical_net_delay]
   [-scalar] [-variation]
   [-aocvm_guardband] [-pocvm_guardband] [-pocvm_coefficient_scale_factor]
   [-increment]
   object_list
```

Data Types

```
object_list      list
```

ARGUMENTS

-hierarchical_net_delay

Resets net delay derating (as well as cell delay derating) previously set by the **set_timing_derate** command for the hierarchical cells specified in the *object_list*. Without the **-hierarchical_net_delay** option, the **reset_timing_derate ... [get_cells ...]** command resets only the cell delay (not net delay) derating in the hierarchical cells.

-scalar

Resets derating previously set by the **set_timing_derate -scalar** ... command. This option should be used only with the PrimeTime VX tool.

-variation

Resets derating previously set by the **set_timing_derate -variation** ... command. This option should be used only with the PrimeTime VX tool.

-aocvm_guardband

Resets derating previously set by the **set_timing_derate -aocvm_guardband** ... command.

-pocvm_guardband

Resets derating previously set by the **set_timing_derate -pocvm_guardband** ... command.

-pocvm_coefficient_scale_factor

Resets derating previously set by the **set_timing_derate -pocvm_coefficient_scale_factor** ... command.

-increment

Resets incremental derating adjustments previously set by the **set_timing_derate -increment** … command.

object_list

Resets the derating previously set for the given list or collection of cells, library cells, or nets.

DESCRIPTION

The **reset_timing_derate** command resets the derating previously set by the **set_timing_derate** command, which effectively changes derating factors back to their original default values (1.0 for delay or constraint derating, or 0.0 for **-increment** derating adjustments).

To reset all derating, including both global and object-specific derating settings, use the **reset_timing_derate** command by itself, without any options.

To reset only some types of derating, use the same scope-setting options as the original **set_timing_derate** command:

```
-aocvm_guardband
-pocvm_guardband
-pocvm_coefficient_scale_factor
-increment
object_list
```

By default, using the **reset_timing_derate** command and specifying a hierarchical cell in the *object_list* causes only the cell delay derating to be reset. To also reset the net derating, use the **-hierarchical_net_delay** option in the command.

If the *object_list* option contains a design, only global derating factors are reset, and object-specific derating factors are retained. For example, to reset all global derating but keep derating set on specific cells and nets, use the following command:

```
pt_shell> reset_timing_derate [current_design]
```

To remove the derating on a subset of an existing derating setting, do not use the **reset_timing_derate** command. Instead, set a derating factor of 1.0 on the subset using another **set_timing_derate**, specifying the more narrow condition (**-net_delay**, **-rise**, and so on) or more specific *object_list*.

EXAMPLES

The following command resets all derating, including both global and instance-specific settings.

```
pt_shell> reset_timing_derate
```

The following command resets all incremental derating adjustments previously set the **set_timing_derate -increment** … commands, including both global and instance-specific settings:

```
pt_shell> reset_timing_derate -increment
```

The following command resets all global derating set on the design MY_DESIGN, without affecting the derating set on specific objects in the design (cells, library cells, and nets).

```
pt_shell> reset_timing_derate [get_design MY_DESIGN]
```

The following example resets all derating previously set on cell U1.

```
pt_shell> reset_timing_derate [get_cell U1]
```

The following example resets all derating previously set on net n123.

```
pt_shell> reset_timing_derate [get_net n123]
```

The following command resets all derating previously set on cells matching "U2*" and on all nets matching "n1*".

```
pt_shell> reset_timing_derate \
  [add_to_collection [get_cells U2*] [get_nets n1*]]
```

The following example resets derating previously set on instances of library cell IV1A from the library MY_LIB.

```
pt_shell> reset_timing_derate [get_lib_cells MY_LIB/IV1A]
```

The following command resets the derating previously set on leaf-level cell H1/U1. As a result, cell U1 inherits the derating set on its parent cell, H1.

```
pt_shell> reset_timing_derate [get_cells H1/U1]
```

The following command resets the cell delay derating previously set on hierarchical cell H1/H2. As a result, cell H2 and all of its lower-level cells inherit the cell delay derating set on their parent (or grandparent) cell, H1.

```
pt_shell> reset_timing_derate [get_cells H1/H2]
```

The following command resets the cell **and net** delay derating previously set on hierarchical cell H1/H2. As a result, cell H2 and all of its lower-level cells inherit the cell **and net** delay derating set on their parent (or grandparent) cell, H1.

```
pt_shell> reset_timing_derate -hierarchical_net_delay [get_cells H1/H2]
```

SEE ALSO

```
set_timing_derate(2)
report_design(2)
report_timing_derate(2)
```

retrieve_rail_result

Retrieves an existing rail result that was created in a previous or the current session.

SYNTAX

```
status retrieve_rail_result
   [name]
```

Data Types

```
string name
```

ARGUMENTS

name

Specifies the name of an existing rail result to be reloaded and set as the current rail result.

DESCRIPTION

The **retrieve_rail_result** command reloads an existing rail result and sets it as current. The existing rail result is generated either during a previous or the current session. You must specify which existing result to be reloaded.

A rail result is automatically created whenever a rail analysis calculation command, such as **calculate_rail_voltage** or **calculate_minimum_path_resistances**, is successfully run. The rail result contains the data generated during the calculation, such as a snapshot of the taps that are created in the session (see **tap_contexts** for more information on tap contexts). PrimeRail reads data from the current rail result when you query the *voltage_drop* attribute of an instance's PG pin, or examine the voltage drop map in the GUI. Therefore, if you want to view or query data that was created during a previous session, you need to first use the **retrive_rail_result** command to retrieve and set this rail result as current.

Unlike the **current_rail_result** command which can retrieve a rail result that is created in the same session, the **retrieve_rail_result** command reloads an existing rail result that is created in a different and sets it as current. Moreover, this the **retrieve_rail_result** command checks if the needed power and ground net extraction data is loaded in the memory. If not, the command loads the needed power and ground extraction data automatically from disk. If the power and ground extraction data is missing or changed after the rail result is being saved, this command run will fail and an error message is issued.

EXAMPLES

The following example retrieves an existing rail result and sets it as current.

```
pr_shell> retrieve_rail_result MyResult_1
1
```

SEE ALSO

calculate_minimum_path_resistances(2)
calculate_rail_voltage(2)
report_rail_results(2)
current_rail_result(2)

save_mw_cel

Saves the specified design in Milkyway format.

SYNTAX

```
status save_mw_cel
   [-as name]
   [-overwrite | -increase_version]
   [-scenarios scenario_list]
   [-hierarchy]
   [-check_only]
   design_name
```

Data Types

```
name            string
scenario_list   list
design_name     string
```

ARGUMENTS

-as *name*

Saves the specified design as a Milkyway design with the specified name. This can be a new or existing Milkyway design.

The name can be formatted as *A*[.*B*], in which the string *A* represents the base name of the design and the string **B** represents the view name of the design: CEL, FRAM, err, or FILL. Refer to the man page of mw_cel_collection for detailed information on specifying a Milkyway design name.

-overwrite

Overwrites the stored version of the Milkyway design. This is the default behavior if the **-increase_version** option is not used.

-increase_version

Increases the version number of the Milkyway design being saved, which preserves the previously stored version. This option can be used together with the **-as** option.

-scenarios *scenario_list*

Specifies a list of scenarios whose constraints are saved with the Milkyway design. By default, the command includes all scenarios.

-hierarchy

Saves the modified top-level Milkyway design, its modified subdesigns, and all attached files. The saved data

overwrites the previously stored versions of the designs. By default, the command saves only the specified design, not its subdesigns.

-check_only

Reports the list of Milkyway designs to be saved, but does not execute the save operation. This option works only with the **-hierarchy** option.

design_name

Specifies explicitly the Milkyway design to be saved. By default, the command saves the current Milkyway design.

DESCRIPTION

This command saves the current Milkyway design or a specified design in Milkyway format. It does not affect the open status of Milkyway designs in the current session. To open or close Milkyway designs, use the **open_mw_cel** or **close_mw_cel** command.

When you use the **-as** option, the command saves the specified design as a new design, along with the files associcated with the design. The associated views overwrite any identically named views if the view name of the saved design is **CEL** and the period character (.) is not specified in the name of the new design.

The **-increase_version** option increases the version of the design to be saved. If you combine the **-as** option with the **-increase_version** option, the command increases the versions of the associated views if the view name of the saved design is **CEL** and the period character (.) is not specified in the name of the design to be saved.

The tool runs a database consistency checker before it saves the Milkyway design. If the check fails, it appends "_CORRUPT" to the Milkyway design name. For example, if you are saving the design named abc.CEL and the consistency checker detects an error, the command writes out the Milkyway design using the name "abc_CORRUPT.CEL".

To save the Milkyway design hierarchically, use the **-hierarchy** option, which saves the lower-level designs in addition to the specified design. When the command finishes saving the design hierarchy, it reports the names of the saved Milkyway views.

The command saves the top-level Milkyway design or subdesigns when they have been modified. When the top-level Milkyway design is unmodified while subdesigns are modified, the commands saves the modified subdesigns and updates the time stamp of the top-level Milkyway design.

The **-hierarchy** option does not close any Milkyway designs. Thus, any open Milkyway designs remain open after the hierarchical save. If you do another hierarchical save, the whole process is applied again for the specified top-level Milkyway design.

When you use the **-check_only** option together with the **-hierarchy** option, the command reports the list of Milkyway designs to be saved, but does not save the Milkyway designs.

Multicorner-Multimode Support

By default, this command uses information from both active and inactive scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example saves the current design as a Milkyway design named "my_new_design".

```
prompt> save_mw_cel -as my_new_design
Information: Performing CEL netlist consistency check. (MWDC-118)
Information: CEL consistency check PASSED. (MWDC-119)
Information: Saved design named my_new_design. (UIG-5)
1
```

The following example saves the current design as a Milkyway design containing consistency errors.n

```
prompt> save_mw_cel -as my_new_design
Information: Performing CEL netlist consistency check. (MWDC-118)
Error: Cell UPC_BLK (0xa0b) reachable from parent hierarchy alu (0xa01)
multiple times. (MWDC-131)
Error: CEL consistency check FAILED (MWDC-100)
Information: Saved design named my_new_design_CORRUPT. (UIG-5)
1
```

The following example saves the "ORCA" design and the designs in its hierarchy.

```
prompt> save_mw_cel -hierarchy ORCA
Starting hierarchical save ...
Saving the cell: BLENDER_4.CEL;1 ...
Saving the cell: BLENDER_1.CEL;1 ...
Saving the cell: BLENDER_3.CEL;1 ...
Saving the cell: BLENDER_0.CEL;1 ...
Saving the cell: ORCA.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
```

In the following example, the designs named top, middle, and bottom are located in different libraries. The command saves the top design and the designs in its hierarchy.

```
prompt> save_mw_cel -hierarchy top
Saving the cell: bottom.CEL;1 from library /work/bottom_lib ...
Saving the cell: middle.CEL;1 from library /work/middle_lib ...
Saving the cell: top.CEL;1 ...
Information: Saved design named top.CEL;1. (UIG-5)
1
```

The following example checks the current design and finds that the BLENDER_0 subdesign has been modified. The next command saves the BLENDER_0 design and the current top-level design, ORCA.

```
prompt> save_mw_cel -hierarchy -check_only
Starting hierarchical save with check only enabled ...
check only -- will save the cell: BLENDER_0.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
prompt> save_mw_cel -hierarchy
Starting hierarchical save ...
Saving the cell: BLENDER_0.CEL;1 ...
Saving the cell: ORCA.CEL;1 ...
Information: Saved design named ORCA.CEL;1. (UIG-5)
1
```

SEE ALSO

```
close_mw_cel(2)
current_mw_cel(2)
open_mw_cel(2)
mw_cel_collection(2)
mw_support_hier_fill_view(3)
```

save_qtm_model

Saves the current quick timing model (QTM) description.

SYNTAX

```
string save_qtm_model
   [-format db | lib]
   [-output file_name]
   [-library_cell]
```

Data Types

```
file_name     string
```

ARGUMENTS

-format db | lib

Specifies the output format to be used. You can specify **db** (the default) or **lib**.

-output *file_name*

Specifies the name of the output file. By default, the tool writes the db file to *model_name*_lib.db. If you use the **-output** option, the tool writes the db file to *output_file*_lib.db file.

-library_cell

Writes the model as a library cell instead of a wrapper design and core library cell. If you use this option, no wrapper design is written, and the model in the *output_file*_lib.db file is named *model_name*.

If you do not specify this option, a wrapper design is written to the *output_file*.db file, and the model in the *output_file*_lib.db file is named *model_name*_core.

DESCRIPTION

This command saves the quick timing model and indicates that the model definition is complete. You must issue all quick timing model commands between the **create_qtm_model** and **save_qtm_model** commands.

To display information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following example saves a wrapper and core quick timing model as two files:

```
pt_shell> save_qtm_model -format db -output myfile
```

The preceding command generates these files:

- myfile_lib.db - Contains the *model_name*_core library cell
- myfile.db - Contains the *model_name* wrapper design

The following command saves the quick timing model as the *model_name* library cell in the myfile_lib.db file:

```
pt_shell> save_qtm_model -format db -output myfile -library_cell
```

The following example saves the quick timing model as a library cell in .lib format and uses the string "current" for the base file name.

```
pt_shell> save_qtm_model -format lib -output current -library_cell
```

SEE ALSO

```
create_qtm_model(2)
report_qtm_model(2)
```

scale_parasitics

Scales the parasitic resistance and capacitance values in memory by specified factors.

SYNTAX

```
status scale_parasitics
   [-resistance_factor r_factor]
   [-ground_capacitance_factor c_factor]
   [-coupling_capacitance_factor cc_factor]
   [-dont_apply_to_annotated]
   [net_list]
```

Data Types

```
r_factor          float
c_factor          float
cc_factor         float
net_list          list
```

ARGUMENTS

-resistance_factor *r_factor*

The scaling factor for parasitic resistance values, a positive floating-point number. For example, a factor of 1.10 increases the resistor values by 10 percent.

-ground_capacitance_factor *c_factor*

The scaling factor for net-to-ground capacitance values.

-coupling_capacitance_factor *cc_factor*

The scaling factor for cross-coupling capacitance values.

-dont_apply_to_annotated

Applies the specified global scaling factors only to parasitic data read in after execution of the **scale_parasitics** command, without affecting parasitic data already annotated on the design. By default, global scaling factors apply to both existing and new parasitic data. This option can be used only to specify scaling factors globally, without the *net_list* option.

net_list

Applies the scaling only to the specified list of nets.

DESCRIPTION

The **scale_parasitics** command scales parasitic resistance and capacitance values by specified factors. You can perform a single parasitic extraction run and then use this command to scale the parasitic data for variation effects, such as resistance dependence on temperature.

The command works for detailed parasitics as well as pi models. The command works irrespective of the source of the parasitic data (SPEF, GPD, DSPF, RSPF, and so on). Any subsequent delay calculations use the scaled parasitics instead of the original parasitics. If you write out parasitics using the **write_parasitics** command, the scaled parasitics are written out.

You can scale the parasitics for the whole design or just a specific list of nets. A new scaling setting overwrites any previous conflicting setting. If you scale the whole design and then a specific net, the net-specific scaling applies to the original value, not the globally scaled value.

If you read in parasitics again for previously scaled nets, the global scaling factors still apply, but net-specific scaling factors are discarded.

To report or cancel scaling factors, use the **report_scale_parasitics** or **reset_scale_parasitics** command.

EXAMPLES

The following example scales the resistance values by a factor 1.1, the ground capacitance values by a factor of 1.3, and the coupling capacitance values by a factor of 0.95. This scaling applies to the whole design and to any new parasitics read in later.

```
pt_shell> scale_parasitics -resistance_factor 1.1 \
          -ground_capacitance_factor 1.3 \
          -coupling_capacitance_factor 0.95
```

The following example scales the ground capacitance by a factor of 1.1 and the coupling capacitance by a factor of 0.95 for net n123. This scaling overwrites any previous global or net-specific setting for the net. If you read new parasitics for the net, the net-specific settings are discarded.

```
pt_shell> scale_parasitics -ground_capacitance_factor 1.1 \
          -coupling_capacitance_factor 0.95 [get_nets n123]
```

SEE ALSO

read_parasitics(2)
reset_scale_parasitics(2)
report_scale_parasitics(2)
write_parasitics(2)

scale_rail_power

Scales power values and current waveforms for the PG pins of the cells to be used in rail analysis.

SYNTAX

```
status scale_rail_power
   [-target_power power_value]
   [-scale_factor scale_value]
   [-type cell_type]
   [cell_list]
   [-exclude_leakage]
   [-include_annotated_power]
   [-include_scaled_power]
   [-reuse]
```

Data Types

```
float power_value
float scale_value
string cell_type
list cell_list
```

ARGUMENTS

-target_power *power_value*

Defines the total power (in Watts) of the cells derived from the -type option or *cell_list*.

-scale_factor *scale_value*

Defines the scale factor applied to every cell derived from the -type option or *cell_list*.

-type *cell_type*

Specifies the type of cells whose power values and current waveforms are to be scaled. Set *cell_type* to *non_clock_network* to scale power for all the cells outside the clock network. Set *cell_type* to *all* to scale power for all the cells.

cell_list

Specifies a list of cells.

-exclude_leakage

Indicates not to scale leakage power and current values. By default, the leakage power and current will be scaled.

-include_annotated_power

Scales the power value that was assigned by the **set_rail_power** command.

-include_scaled_power

Scales the power value that was scaled by the previous **scale_rail_power** commands.

-reuse

Indicates to read in **scale_rail_power** from a previous session. User scaled power from **scale_rail_power** commands in a previous session is saved to RAIL_DATABASE after a successful run of **calculate_rail_voltage** from the last session.

This option is mutually exclusive with other options in the command.

DESCRIPTION

The **scale_rail_power** command scales power values and current waveforms to cell PG pins. The cell PG pins are specified by either cell list or cell type. The command returns 1 if successful, and 0 if unsuccessful.

Run the **update_currents** command to generate power and current waveforms before scaling power using the **scale_rail_power** command. The power value and current waveform calculated by the **update_currents** command are scaled accordingly. If you rerun the **update_currents** command, the generated power value and current waveforms overwrite the previously scaled power and current waveforms.

The commands **set_rail_power** and **set_rail_current** have higher priority than the **scale_rail_power** command. The power value set by the **set_rail_power** command cannot be scaled by default (but can be scaled if the -include_annotated_power option is specified), and the current waveform set by the **set_rail_current** command cannot be scaled. The power values and current waveforms of the remaining cell instances will be scaled based on the setting of the -target_power or -scale_factor option.

After the scaling is performed, the attribute **has_scaled_power** is set to true on the cells and the pg_pins of the cells for which the power was scaled. The **total_power** attribute on the respective cells and pg_pins reflects the newly scaled power value.

EXAMPLES

The following example scales the total power for cell instances U2 and U3 to 4.0.

```
pr_shell> update_currents -static
pr_shell> set_rail_power 1.0 [get_cell U1]
pr_shell> scale_rail_power -target_power 5.0 {U1 U2 U3}
```

The following example scales the power and current waveforms by half for every cell instance that is not on the clock network. The leakage power and current values remain unchanged.

```
pr_shell> update_currents -dynamic
pr_shell> scale_rail_power -scale_factor .5 -type non_clock_network -exclude_leakage
```

SEE ALSO

```
set_rail_power(2)
set_rail_current(2)
update_currents(2)
```

set_active_clocks

Sets a group of clocks to be active in the current analysis scope.

SYNTAX

```
status set_active_clocks
   active_clock_list
   [-all_clocks]
```

Data Types

```
active_clock_list        list
```

ARGUMENTS

active_clock_list

Specifies a list of clocks matching the clock names. The *active_clock_list* and **all_clocks** are mutually exclusive.

-all_clocks

Specifies to analyze all clocks simultaneously. The *active_clock_list* and **all_clocks** are mutually exclusive.

DESCRIPTION

Sets a group of clocks to be active in the current analysis scope. Provides capability to perform timing analysis for a particular set of clock domains, but not the entire chip. If a few of the clocks are set as active, the tool performs timing analysis only for these clock domains.

If this command is not specified, the tool analyzes all clocks. The last **set_active_clocks** command always overrides previous ones. If you want to analyze all clocks simultaneously again, use the **set_active_clocks** command with the *active_clock_list* option set to a value of * or use it with the **all_clocks**.

When a clock or a generated clock is created, it is active by default. If a generated clock is active itself, but its master clock is inactive, it is not analyzed. You can use the *is_active* attribute of the clock object to create a collection of active clock objects. Thus, you can freely add or remove individual clocks from this collection.

You can use the **set_active_clocks** and **set_clock_groups** commands together to simultaneously analyze multiple clocks per register without manually specifying false paths. For examples, see the **set_clock_groups** man page.

When the **set_active_clocks** command applied on crosstalk delay analysis, only the victim and aggressors driven by active clocks are considered. Therefore, the nets driven by the non-active aggressor are considered

quiet. However, all nets are sensitive to static noise (as victim) even when there is no active clocks driving them. This feature could be used to set test clocks to non-active when the design is in normal or mission mode and vice versa. It is important that you set all active clocks in the design as active, as they can be aggressors to other part of the design.

This command is not supported with the **write_sdc**, **characterize_context**, and **extract_model** commands.

The **set_active_clocks** command is used to speed up update_timing for local analysis, and some global functionalities might not behave as you expect. For example, check_timing reports endpoints as unconstrained if they are captured by inactive clocks.

EXAMPLES

The following example sets only two of the existing clocks in the design to be active.

```
pt_shell> set_active_clocks {clk1 clk2}
```

The following example resets all clocks in the design to be active.

```
pt_shell> set_active_clocks [all_clocks]
```

The following example adds *clk3* to the current active clocks list.

```
pt_shell> set het [get_clock * -filter is_active==true]
pt_shell> set newfoo [add_to_collection $het [get_clocks clk3]]
pt_shell> set_active_clocks $newhet
```

The following example removes *clk3* from the current active clocks list.

```
pt_shell> set het [get_clock * -filter is_active==true]
pt_shell> set newfoo [remove_from_collection $het \
[get_clocks * -filter "is_active==true && full_name == clk3"]]
pt_shell> set_active_clocks $newhet
```

SEE ALSO

add_to_collection(2)
all_clocks(2)
create_clock(2)
create_generated_clock(2)
get_clocks(2)
set_clock_groups(2)
remove_clock_groups(2)
remove_from_collection(2)
report_clock(2)

set_annotated_check

Sets the setup, hold, recovery, removal, or nochange timing check value between two pins.

SYNTAX

```
string set_annotated_check
   -setup | -hold | -recovery | -removal | -nochange_high | -nochange_low | -width | -period
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-from from_pins]
   [-to to_pins]
   [-clock rise | fall]
   [-cond sdf_expression]
   [-worst]
   [-increment]
   [-override_increment]
   check_value
```

Data Types

from_pins	list
to_pins	list
sdf_expression	string
check_value	float

ARGUMENTS

-setup

Specifies that data must be stable for the amount of time specified by the *check_value* option before the closing clock edge.

-hold

Specifies that data must be stable for the amount of time specified by the *check_value* option after the closing clock edge.

-recovery

Specifies that an asynchronous set or clear inactive edge cannot occur within the *check_value* option before the closing clock edge. This is essentially a setup requirement on an asynchronous pin, but only the inactive transition is considered.

-removal

Specifies that an asynchronous set or clear inactive edge cannot occur until the *check_value* option time after

the closing clock edge. This is essentially a hold requirement on an asynchronous pin, but only the inactive transition is considered.

-nochange_high

Specifies that the data must not rise within the *check_value* option time before the clock becomes active, and must not fall until the *check_value* variable time after the clock becomes inactive. A rise data transition corresponds to the check before the activating clock edge. A fall data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active high. A fall clock transition indicates that the clock is active low.

-nochange_low

Specifies that the data must not fall within the *check_value* option time before the clock becomes active, and must not rise until the *check_value* option time after the clock becomes inactive. A fall data transition corresponds to the check before the activating clock edge. A rise data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active high. A fall clock transition indicates that the clock is active low.

-width

Specifies the minimum pulse width constraint for a clock pin. The timing check is defined on the **-from** pin, so the related pin (**-to**) should either be the same or it can be omitted.

-period

Specifies the minimum period constraint for a clock pin. The timing check is defined on the **-from** pin, so the related pin (**-to**) should either be the same or it can be omitted.

-rise

Specifies that the timing check is for the data rise transition. If you do not specify either the **-rise** or **-fall** options, both values are set.

-fall

Specifies that the timing check is for the data fall transition. If you do not specify either the **-rise** or **-fall** options, both values are set.

-min

Use this option only if the design is in min_max mode (min and max operating conditions). Specifies the minimum timing check for both data rise and data fall transitions.

-max

Use this option only if the design is in min_max mode (min and max operating conditions). Specifies the maximum timing check for both data rise and data fall transitions.

-from *from_pins*

Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs for which checks are annotated. There must be a corresponding timing arc between the *from_pins* and the *to_pins* options.

-to *to_pins*

Specifies a list of leaf cell data pins that are the endpoints of the timing arcs for which checks are annotated. There must be a corresponding timing arc between the *from_pins* and the *to_pins* options.

-clock rise | fall

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are set.

-cond *sdf_expression*

Use this option only if the library has a condition attached to the specified timing check arc; otherwise, an error message is generated. Specifies the condition for which the annotated check is valid. The syntax of the condition must match the condition specified in the library using the sdf_cond construct. The syntax is the same one used in the Standard Delay Format (SDF).

-worst

This option is not currently implemented, so it is ignored.

-increment

Specifies that the delay value is to be added to the calculated check value of the specified timing arc. If an annotated incremental value already exist, then the new value will be added on the existing one.

-override_increment

Overwrite incremental values instead of adding them on the already existing. Applies only for annotating incremental values.

check_value

Specifies the timing check value between pins on the same cell, in units consistent with the logic library used during optimization. For example, if the logic library specifies delay values in nanoseconds, the *check_value* option must be expressed in nanoseconds. The *check_value* option can be negative.

DESCRIPTION

Annotates a timing check arc between two or more pins on a cell or a net in the current design. This can be appropriate after place and route, for technologies where the timing check value varies for different instances, and the library timing check values do not provide sufficient accuracy.

If the design is not linked, it is linked automatically by the **set_annotated_check** command.

You can use the **set_annotated_check** command for pins at lower levels of the design hierarchy. You can specify pins in the format of *INSTANCE1/INSTANCE2/PIN_NAME*.

To list annotated timing check values, use the **report_annotated_check** command. To remove annotated timing check values from a design, use the **remove_annotated_check** or **reset_design** commands. To see the effect of the **set_annotated_check** command for a specific instance, use the **report_timing** command.

EXAMPLES

The following example annotates a setup time of 2.1 units between the CP clock pin of cell instance u1/ff12 and the D data pin of the same cell instance.

```
pt_shell> set_annotated_check -setup 2.1 -from u1/ff12/CP -to u1/ff12/D
```

The following example annotates a check for the maximum value of 3.0 units for a minimum pulse width of a clock signal in 'high' state at the u1/ff12/CP clock pin. The annotation is then reported by **report_annotated_check**:

```
pt_shell> set_annotated_check -width 3.0 -clock rise -max  -from u1/ff12/CP
```

```
pt_shell> report_annotated_check -width -list_annotated
****************************************
Report : annotated_check
        -width
        -list_annotated
 ...
****************************************

Backannotated cell min_pulse_width check arcs:
----------------------------------------------
1. u1/ff12/CP -> u1/ff12/CP  (r:0.10/3.00  f:NA  sense: clock_pulse_width_high)

                              |           |           |    NOT    |
    Timing check type         |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell min pulse width arcs     |       20  |        1  |       19  |
```

The following examples illustrates how the incremental values are annotated and how incremental values are used.

On the example below a value of 1.0 unit is annotated on the setup time between the clock pin and the data pin of a cell instance. After that an incremental setup value of 0.5 units is annotated on the same arc. The final value of the setup constraint on this arcs is 1.5.

```
pt_shell> set_annotated_check -setup -from ff/CP -to ff/D 1.0
pt_shell> set_annotated_check -increment -setup -from ff/CP -to ff/D 0.5
```

Continuing the above example, the following command annotates another 0.6 unit on this setup arc. The new setup value for this arc is now 2.1, where 1.0 is the fully annotated value and 1.1 (0.5 + 0.6) is the incremental annotated value.

```
pt_shell> set_annotated_check -increment -setup -from ff/CP -to ff/D 0.6
```

The overriding of the previously annotated incremental values on a timing arc can be achieved with option **-override_increment**. The impact that the following command has on the above example is to override the current increment value of 1.1 with a new one of 0.7. The new setup value is now 1.7 since 1.0 was the initial fully annotated value and 0.7 in the new increment value.

```
pt_shell> set_annotated_check -increment -override_increment -setup -from ff/CP -to ff/D 0.7
```

The incremental annotated values equally interacts with fully annotated values and calculated values. They can also be removed with the usage of the **remove_annotated_check** or **reset_design** commands.

SEE ALSO

```
current_design(2)
link(2)
read_sdf(2)
remove_annotated_check(2)
report_annotated_check(2)
report_timing(2)
reset_design(2)
```

set_annotated_clock_network_power

Annotate power on clock networks.

SYNTAX

```
string set_annotated_clock_network_power
   [-internal_power internal_power]
   [-switching_power switching_power]
   [-leakage_power leak_power]
   [-total_power total_power]
   [-clock domain_clock]
```

Data Types

internal_power	float
switching_power	float
leak_power	float
total_power	float
domain_clock	string

ARGUMENTS

-internal_power *internal_power*

Specifies the annotated internal power value on the clock network.

-switching_power *switching_power*

Specifies the annotated switching power value on the clock network.

-leakage_power *leak_power*

Specifies the annotated leakage power value on the clock network.

-total_power *total_power*

Specifies the annotated total power value on the clock network. It is treated as internal power in the reports.

-clock *domain_clock*

Specifies the clock in the related clock network on which the power values are annotated.

DESCRIPTION

The **set_annotated_clock_network_power** command provide the capability to annotate the power values on the clock networks in the reports from PrimeTime PX. If this command is used before the **report_power** command, PrimeTime PX uses the annotated clock network power values in the summary report. There is also a separate annotated clock network power report embedded in the regular summary power report. The *clock_network* group power in the summary report excludes the clock network objects whose power values are annotated. An attribute 'e' marks such a situation in the report.

The power values in the unit of Watt are specified by either *-total_power* option of a combination of *-internal_power*, *-switching_power* and *-leakage_power* options, If *-total_power* option is used, the power values are also treated as internal power in the reports to make the results consistent. If the options of *-switching_power*, *-internal_power* and/or *-leakage_power* are used, the values are annotated on internal, switching and/or leakage power respectively. The total power is the sum of these three components. These three options can be used together or separately. But they cannot be used together with the *-total_power* option; otherwise, the CMD-001 error message is issued and the command fails.

By default, the power values aree annotated on the whole clock network in the design. If *-clock* option is used, the power values are annotated on the collection of clock network objects of the corresponding clock domain. Only one clock is allowed for one command. Multiple **set_annotated_clock_network_power** commands can be used to specify annotated power of multiple clock domains separately. The annotated clock network power reports list them separately.

If two **set_annotated_clock_network_power** commands are specified for the same clock domain (or the whole clock network) and the power types (internal, switching, leakage or total) are the same, The power values of the later command overrides the earlier one. If they are not the same power type, both power values are kept and reported.

The **set_annotated_clock_network_power** command can work with **report_power -groups group_list** only when the group_list contains the default 'clock_network' power group; otherwise, an PWR-296 error message is issued and the **report_power** command fails.

The **set_annotated_clock_network_power** command causes the cells in the related clock network not to be reported by **report_power -cell** command. The other options in the **report_power** command are not affected.

The annotated clock network power has higher priority than the estimated clock network power. If both features are invoked in the **report_power** command, a PWR-295 warning message is issued and annotated clock network power values are used in power reports.

To remove the annotated clock network power values already specified for the design, use the **remove_annotated_clock_network_power** command. The **report_power** command reverts back to its original behavior.

EXAMPLES

In the following example, the internal, switching power are annotated separately for clock domain CKA and CKB.

```
pt_shell> create_clock -name CKA -period 10 clka
pt_shell> create_clock -name CKB -period 5 clka
pt_shell> set_annotated_clock_network_power -internal 1.0e-03 -switching 2.0e-03 -clock CKA
pt_shell> set_annotated_clock_network_power -switching 2.0e-03 -clock CKB
pt_shell> report_power
```

The following is the output from the above commands

```
****************************************
Report : Statistical Average Power
Design : my_design
```

```
Version: my_version
Date   : ....
****************************************

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group
      e  -  Annotated clock network power excluded

                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power   (%)  Attrs
--------------------------------------------------------------------------------
io_pad                     0.0000    0.0000    0.0000    0.0000 (0.00%)
memory                     0.0000    0.0000    0.0000    0.0000 (0.00%)
black_box                  0.0000    0.0000    0.0000    0.0000 (0.00%)
clock_network           3.536e-03    0.0000    0.0000 3.536e-03 (36.81%)  ei
register                3.060e-03    0.0000 1.020e-05 3.071e-03 (31.96%)
combinational              0.0000    0.0000    0.0000    0.0000 (0.00%)
sequential                 0.0000    0.0000    0.0000    0.0000 (0.00%)


                        Internal  Switching  Leakage    Total
Clock                   Power     Power      Power      Power   (%)  Attrs
--------------------------------------------------------------------------------
CKB                     0.0000     2.000e-03  0.0000    2.000e-03
CKA                     1.000e-03  0.0000     0.0000    1.000e-03
--------------------------------------------------------------------------------
All Annotated Clocks    1.000e-03  2.000e-03  0.0000    3.000e-03 (31.23%)


  Net Switching Power  = 2.000e-03   (20.82%)
  Cell Internal Power  = 7.596e-03   (79.07%)
  Cell Leakage Power   = 1.020e-05   ( 0.11%)
                         ---------
Total Power            = 9.607e-03  (100.00%)
```

SEE ALSO

remove_annotated_clock_network_power(2)
report_power(2)

set_annotated_delay

Sets the net or cell delay value between two pins.

SYNTAX

```
string set_annotated_delay
   -cell | -net
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-load_delay net | cell]
   [-from from_pins]
   [-to to_pins]
   [-cond sdf_expression]
   [-increment]
   [-delta_only]
   [-worst]
   [-variation variation_object]
   [-of_objects objects]
   delay_value
```

Data Types

from_pins	list
to_pins	list
sdf_expression	string
variation_object	string
objects	list
delay_value	float

ARGUMENTS

-cell

Specifies that the annotated delay is a cell delay. You cannot use the **-cell** option together with the **-net** option.

-net

Specifies that the annotated delay is a net delay. You cannot use the **-net** option together with the **-cell** option.

-rise

Specifies that the delay is for the data rise transition. If you do not specify either the **-rise** or **-fall** option, both values are set.

-fall

Specifies that the timing check is for the data fall transition. If you do not specify either the **-rise** or **-fall** option, both values are set.

-min

Specifies the minimum delay for both data rise and data fall transitions. Use this option only if the design is in min_max mode (min and max operating conditions).

-max

Specifies the maximum delay for both data rise and data fall transitions. Use this option only if the design is in min_max mode (min and max operating conditions).

-load_delay net | cell

Specifies one of the following types of load delay:

- **net** - Includes the load delay as part of the annotated net delays.
- **cell** - Includes the load delay as part of the annotated cell delays.

Load delay is the portion of cell delay resulting from the capacitive load of the net the cell is driving. All timing arcs of the same net and of the same cell must be annotated with the same type of load delay.

-from *from_pins*

Specifies a list of leaf-cell pins and top-level ports that are the startpoints of the timing arcs for which delays are annotated. The **-from/to** and **-of_objects** options are mutually exclusive; you must specify one, but not both.

-to *to_pins*

Specifies a list of leaf cell pins or top-level ports that are the endpoints of the timing arcs for which delays are annotated. The **-from/to** and **-of_objects** options are mutually exclusive; you must specify one, but not both.

-cond *sdf_expression*

Specifies the condition for which the annotated delay is valid. The syntax of the condition must match the condition specified in the library using the construct sdf_cond. The syntax is the same one used in the Standard Delay Format (SDF). Use this option only if the library has a condition attached to the specified delay timing arc; otherwise, an error message is generated.

-increment

Increments the delay to the current delay of the specified timing arc.

-delta_only

Adds the annotated delay to the net delay value calculated by the tool. You cannot use this option with the **-cell** option.

-worst

This option is not used by PrimeTime, so it is ignored.

-variation *variation_object*

Specifies a variation to annotate on all arcs between the from and to pins.

-of_objects *objects*

Specifies a list of timing arcs (created with the **get_timing_arcs** command) on which to annotate. The **-of_objects** and **-from/to** options are mutually exclusive; you must specify one, but not both.

delay_value

Specifies the delay value between pins on the same cell, in units consistent with the logic library used during optimization. For example, if the logic library specifies delay values in nanoseconds, *delay_value* must be expressed in nanoseconds.

DESCRIPTION

This command annotates cell or net delays in the design.

Cell delays exist between pins of the same leaf cell. With the **-cell** option, pins in *from_list* must be cell input or inout pins, and pins in *to_list* must be cell output or inout pins.

Net delays exist between leaf-cell pins or top-level ports connected by a net. With the **-net** option, pins in *from_list* must be cell output or inout pins or top-level input ports, and pins in *to_list* must be cell input or inout pins or top-level output ports.

To annotate the load-dependent delay due to driving cells on input ports, use the **-cell** option, omit the **-from** option, and specify the input ports with the **-to** option.

To verify that the back-annotation done by **set_annotated_delay** is correct, run the **update_timing** command.

Load delay, also known as extra source gate delay, is the portion of cell delay caused by the capacitive load of the driven net. Some delay calculators consider load delay part of the net delay and others consider it part of the cell delay. If your annotated delay value (for either cell or net) assumes that load delay is part of the cell delay, use the **-load_delay cell** option. If your delay value assumes that load delay is included in the net delay, use the **-load_delay net** option. By default, load delays are assumed to be in cell delays, so they appear in the **report_timing** path listings.

The specified delay value overrides the internally-estimated cell and net delay value. If the specified pins are not in the same cell or on the same net, when the timing is updated an error message is generated and *delay_value* is discarded for those pins.

If a timing arc has annotated delta delay (**set_annotated_delay -net -delta_only**) and then total arc delay is annotated (**set_annotated_delay -net**), the arc delay is assumed to include the delta. In other words, delta delay is not used if delay is annotated. The annotated delta delay becomes significant after the annotated delay is removed.

The **set_annotated_delay** command can be used for pins at lower levels of the design hierarchy. Pins are specified in the format of INSTANCE1/INSTANCE2/PIN_NAME.

To list annotated delay values, use the **report_annotated_delay** command. To remove the annotated cell or net delay values from a design, use the **remove_annotated_delay** or **reset_design** command.

EXAMPLES

The following example annotates a cell delay of 20 units between input pin A of cell instance U1/U2/U3 and output pin Z of the same cell instance. The delay value of 20 includes the load delay.

```
pt_shell> set_annotated_delay -cell -load_delay cell 20 \
          -from U1/U2/U3/A -to U1/U2/U3/Z
```

The following example annotates a rise net delay of 1.4 units between output pin U1/Z and input pin U2/A. The delay value for this net does not include load delay.

```
pt_shell> set_annotated_delay -net -rise 1.4 -load_delay cell \
          -from U1/Z -to U2/A
```

The following example annotates a rise net delay of 12.3 units between the same output pins. In this example, the net delay value includes load delay.

```
pt_shell> set_annotated_delay -net -rise 12.3 -load_delay net \
          -from U1/Z -to U2/A
```

SEE ALSO

current_design(2)
link(2)
read_sdf(2)
remove_annotated_delay(2)
report_annotated_delay(2)
report_timing(2)
reset_design(2)

set_annotated_mtcmos_switch_cell

Sets the status of the related annotations of the MTCMOS switch cell instance defined by objects.

SYNTAX

```
status set_annotated_mtcmos_switch_cell
   object_list
   -on_state_resistance res_value | -off_state_leakage
   leakage_value | -state on_off_state
   [-pin_state pin_state_list | -pin_state_name pin_state_name]
```

Data Types

```
float res_value
float leakage_value
string on_off_state
list object_list
list pin_state_list
string pin_state_name
```

ARGUMENTS

object_list

Specifies a list of cell instances or library cell objects on which to apply the switch-related user annotation. Each element in the list can be a collection of cell objects or a cell object name pattern.

-on_state_resistance *res_value*

Specifies an on-state resistance value that is annotated on to the MTCMOS switch cell instances defined by the object list. An on-state resistance value cannot be annotated to a pin state when the cell is turned off.

-off_state_leakage *leakage_value*

Specifies an off-state leakage value that is annotated on to the MTCMOS switch cell instances defined by the object list. An off-state leakage value cannot be annotated to a pin state when the cell is turned on.

-state *on_off_state*

Specifies an on- or off-state that is annotated on to the MTCMOS switch cell instances defined by the object list. The state will be ultimately used during any subsequent static rail analysis. The allowed string values are on and off. By default, all the MTCMOS switch cell instances' states are considered to be on for static rail analysis. When the *-state* option is specified, the command checks whether the state is consistent with the case analysis result if there is a case already set during timing or power analysis. The command issues an error message if an MTCMOS switch cell is set off in case analysis during timing or power analysis, but is set on during rail analysis.

When neither the *-pin_state* nor *-pin_state_name* option is specified, the on or off state is applied to all pin states where the cell can be turned on or off, respectively. When only the option setting of *-state on* is used, the pin state which has the smallest on-state resistance becomes the current pin state. Similarly, when only the option *-state off* is used, the pin state which has the smallest off state leakage becomes the current pin state. When either *-pin_state* or *-pin_state_name* is specified, the on or off state is applied to the specified pin state, and the specified pin state becomes the current pin state. If under the specified pin state, the cell cannot be turned on or off and the command will fail.

-pin_state *pin_state_list*

Specifies a list of input pin states for which the annotated value is applied to. If an input pin does not exist in the cell, the command will fail. If the MTCMOS switch cell consists of more input pins than those specified in the pin state list, the command will fail. This option is optional only if either *-state* or the *-pin_state_name* is also specified. In conjunction with the *-state* option, if this option is not specified, the annotation is applied to all pin states of the MTCMOS switch cell in the library. This option cannot be specified with the *-pin_state_name* option.

-pin_state_name *pin_state_name*

Specifies a pin state name for which pin state the annotated value is applied to. If the pin state name does not exist for the library cell that the MTCMOS cell instance is referenced to, the command will fail. This option is optional only if the option *-state* or the option *-pin_state* is specified. In conjunction with the *-state* option, if this option is not specified, the annotation is applied to all pin states of the MTCMOS switch cell. This option cannot be specified with the *-pin_state* option.

DESCRIPTION

This command enables you to set a user annotation for the specified MTCMOS switch cell instance objects. You can set three types of annotations: on-state resistance, off-state leakage, and state. The on-state resistance and off-state leakage are typically included as part of the rail characterization data for the containing MTCMOS switch cell (also known as power management cell), and so is automatically determined when the library is read in for the MTCMOS library cell objects. You can use this command to change the resistance, leakage or on/off state values for a switch cell instance. If an on-state resistance value is not included in the library, the MTCMOS switch cell will not be assumed to be able to connect two PG networks. If an off-state leakage is not included in the library, the default value will be 0.0. The on-state resistance is represented as a *cell* or *lib_cell* attribute, called *on_state_resistance.* The off-state resistance is represented as a *cell* or *lib_cell* attribute, called *off_state_leakage.* Use the **list_attributes -class cell** or **lib_cell -app** attributes if needed.

There are a few notes regarding to usage of this command. First if the library information is not provided, run the **set_mtcmos_switch_lib_cell** command instead. Second, this command can only add an annotation to a cell rather than a library cell. Third, the pin_state specifies a state where the MTCMOS switch cell is turned on fully (in cases when a switch is inside the MTCMOS switch cell) or partially (in cases when multiple switches are inside the MTCMOS switch cell) or turned off fully (when all internal switches inside the MTCMOS switch are controlled to off by the pin state). The tool checks whether the annotated value has any conflict with the pin state. A conflict occurs if an on-state resistance or the state on is annotated to a pin state when the cell is actually off as per the library definition, or similarly an off-state leakage or the state off is annotated to a pin state where the cell is on. If there is no conflict, the annotation will be set to the specified pin state.

A typical usage of this user annotation is for a what-if flow where the resistance values, the leakage values or the on/off states can be adjusted to make voltage drop tradeoffs that simulate MTCMOS switch cell sizing operations without changing the cell references and the design and then quickly iterating with static rail analysis to assess result tradeoffs. See the **calculate_rail_voltage -static** man pages for further details.

EXAMPLES

An MTCMOS cell has been defined in the library, where the cell is on if the control pin sw has an input of a logic zero. The following example sets a user-annotated on-state resistance to the cell such that the user-annotated resistance will be used during rail analysis:

```
pr_shell> set_annotated_mtcmos_switch_cell \
                  -on_state_resistance 100.0 -pin_state {sw 0} \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following example shows a failure of the command run because the cell is off under pin state {sw 1}, and an on-state resistance is to be annotated:

```
pr_shell> set_annotated_mtcmos_switch_cell \
                  -on_state_resistance 100.0 -pin_state {sw 1} \
                  [get_cell -filter is_mtcmos_switch_cell]
```

However, the following example shows a successful command run because the command sets an off-state leakage, and the cell is off under pin state {sw 1}:

```
pr_shell> set_annotated_mtcmos_switch_cell \
                 -off_state_leakage 2.0 -pin_state {sw 1} \
                 [get_cell -filter is_mtcmos_switch_cell]
```

An MTCMOS cell has been defined in the library, where the cell is on if the when condition sw1 || sw2 is satisfied. This MTCMOS cell also consists of two transistor switches controlled by pins sw1 and sw2. The following command sets on-state resistance for all possible pin states which satisfy the library when condition:

```
pr_shell> set_annotated_mtcmos_switch_cell \
                 -on_state_resistance 500.0 [get_cell -filter is_mtcmos_switch_cell]
```

The following command sets on-state resistance for only one of the pin states:

```
pr_shell> set_annotated_mtcmos_switch_cell \
                  -on_state_resistance 400 -pin_state {sw1 1 sw2 0} \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following command turns off the cell without specifying a pin state or pin state name:

```
pr_shell> set_annotated_mtcmos_switch_cell -state off \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following command turns on the cell without specifying a pin state or pin state name:

```
pr_shell> set_annotated_mtcmos_switch_cell -state on \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following command sets the current pin state to be "sw1 1 sw2 0," which means only one switch transistor inside the cell is turned on:

```
pr_shell> set_annotated_mtcmos_switch_cell -state on \
                  -pin_state {sw1 1 sw2 0} \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following command sets the current pin state to be "sw1 0 sw2 0," which means the cell is turned off.

```
pr_shell> set_annotated_mtcmos_switch_cell -state off \
                  -pin_state {sw1 0 sw2 0} \
                  [get_cell -filter is_mtcmos_switch_cell]
```

The following command assigns a pin state name for a pin state and assigns an off-state leakage value, and then turns off the cell for the pin state using the specified pin state name:

```
pr_shell> set_mtcmos_switch_cell -off_state_leakage 1.0 \
                  [get_lib_cell -filter is_mtcmos_switch_cell] \
                  -pin_state {sw1 0 sw2 0} -pin_state_name low
pr_shell> open_mw_cell -lib design cell
pr_shell> set_annotated_mtcmos_switch_cell -state off \
                  -pin_state_name low \
                  [get_cell -filter is_mtcmos_switch_cell]
```

SEE ALSO

```
get_cell(2)
get_lib_cell(2)
remove_annotated_mtcmos_switch_cell(2)
report_annotated_mtcmos_switch_cell(2)
calculate_rail_voltage(2)
set_mtcmos_switch_lib_cell(2)
```

set_annotated_power

Annotates power on unresolved black box cells or leaf cells.

SYNTAX

```
status set_annotated_power
   [-internal_power internal_value]
   [-leakage_power leakage_value]
   [-switching_power switching_power_value]
   [-rails rail_list]
   object_list
```

Data Types

internal_value	float
leakage_value	float
switching_power_value	float
rail_list	list
object_list	list

ARGUMENTS

-internal_power *internal_value*

Specifies the internal power (in watts) to be annotated.

-leakage_power *leakage_value*

Specifies the leakage power (in watts) to be annotated.

-switching_power *switching_power_value*

Specifies the switching power (in watts) to be annotated.

-rails *rail_list*

Specifies a list of rails/power supply nets on which power is annotated. This option applies the annotated power values to each power supply net (or rail in non-UPF mode) in the list. This feature is valid only if the **power_enable_multi_rail_analysis** variable is set to **true**.

object_list

Specifies a list of cells on which to annotate internal_power or leakage_power, or a list of nets on which annotate switching_power.

DESCRIPTION

The **set_annotated_power** command annotates the internal and leakage power on the specified cells and annotates the switching power on the specified nets. The command should be used mostly on unresolved black box cells, where internal and leakage power cannot be estimated. However, the command also works for any leaf cells and hierachical cells, where the annotated internal and leakage power overrides the internally estimated internal and leakage power. The internally estimated power, if available, is overridden by the annotated power number, but is resurrected if the annotated power is removed.

Switching power (power of the nets connected to the output pins of the cell) can be annotated using list of nets.

The annotated power is average power, so it has no effect on power waveforms.

EXAMPLES

The following example shows how power is annotated, removed and reported.

```
pt_shell> set_annotated_power -internal_power 0.1 -leakage_power 0.2 U0/U1
1
pt_shell> set_annotated_power -internal_power 0.3 -leakage_power 0.4 U0
1
pt_shell> set_annotated_power -switching_power .25 out
1
pt_shell> set_annotated_power -switching_power .25  n1

****************************************
Report : annotated_power
        -list_annotated
****************************************

Annotated cell powers:
------------------------------


 1. U0  (internal: 0.3  leakage: 0.4)

 2. U0/U1  (internal: 0.1  leakage: 0.2)



Annotated net powers:
------------------------------


 1. out                    (switching : 0.25       )

 2. n1                     (switching : 0.25       )



                             |           |           |    NOT    |
     Cell type               |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell    |         0 |         0 |         0 |
leaf cell                    |         5 |         1 |         4 |
hierarchical cell            |         1 |         1 |         0 |
```

```
net                           |       11  |        2  |        9  |
------------------------------+-----------+-----------+-----------+
                              |       17  |        4  |       13  |

1
pt_shell> remove_annotated_power n1
1
pt_shell> report_annotated_power -list_annotated
****************************************
Report : annotated_power
****************************************
Annotated cell powers:
------------------------------


 1. U0  (internal: 0.3  leakage: 0.4)

 2. U0/U1  (internal: 0.1  leakage: 0.2)



Annotated net powers:
------------------------------


 1. out                    (switching : 0.25       )



                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
unresolved black-box cell     |        0  |        0  |        0  |
leaf cell                     |        5  |        1  |        4  |
hierarchical cell             |        1  |        1  |        0  |
net                           |       11  |        1  |       10  |
------------------------------+-----------+-----------+-----------+
                              |       17  |        3  |       14  |
1
```

SEE ALSO

remove_annotated_power(2)
report_annotated_power(2)

set_annotated_transition

Sets the transition time annotated on specified pins in the current design.

SYNTAX

```
int set_annotated_transition
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-delta_only]
   slew_value
   pin_list
```

Data Types

```
slew_value     float
pin_list       list
```

ARGUMENTS

-rise

Indicates that the *slew_value* variable represents the data rise transition time.

-fall

Indicates that the *slew_value* variable represents the data fall transition time.

-min

Indicates that the *slew_value* variable represents the minimum transition time. Use this option only if the design is in min-max mode (min and max operating conditions).

-max

Indicates that the *slew_value* variable represents the maximum transition time. Use this option only if the design is in min-max mode (min and max operating conditions).

-delta_only

Indicates that the *slew_value* variable represents a delta transition time that is added to the transition time computed by delay calculation.

slew_value

Specifies the transition time of the specified pins or ports, in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, the *slew_value*

variable must be expressed in nanoseconds. If used with the *-delta_only* option, the *slew_value* variable can be a negative number.

pin_list

Specifies a list of pins or ports that is annotated with the transition time *slew_value*.

DESCRIPTION

Specifies the transition time that is annotated on pins in the current design. The specified transition time value overrides the internally-estimated transition time value.

The **set_annotated_transition** command can be used for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To remove annotated transition times, use the **remove_annotated_transition** command.

EXAMPLES

The following example annotates a rising transition time of 0.5 units and a falling transition time of 0.7 units on input pin **A** of cell "U1/U2/U3".

```
pt_shell> set_annotated_transition -rise 0.5 [get_pins U1/U2/U3/A]
pt_shell> set_annotated_transition -fall 0.7 [get_pins U1/U2/U3/A]
```

SEE ALSO

```
remove_annotated_transition(2)
report_timing(2)
reset_design(2)
set_annotated_delay(2)
```

set_app_var

Sets the value of an application variable.

SYNTAX

```
string set_app_var
   -default
   var
   value
```

Data Types

```
var       string
value     string
```

ARGUMENTS

-default

Resets the variable to its default value.

var

Specifies the application variable to set.

value

Specifies the value to which the variable is to be set.

DESCRIPTION

The **set_app_var** command sets the specified application variable. This command sets the variable to its default value or to a new value you specify.

This command returns the new value of the variable if setting the variable was successful. If the application variable could not be set, then an error is returned indicating the reason for the failure.

Reasons for failure include:

- The specified variable name is not an application variable, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true See the **sh_allow_tcl_with_set_app_var** man page for details.
- The specified application variable is read only.

- The value specified is not a legal value for this application variable

EXAMPLES

The following example attempts to set a read-only application variable:

```
prompt> set_app_var synopsys_root /tmp
Error: can't set "synopsys_root": variable is read-only
       Use error_info for more info. (CMD-013)
```

In this example, the application variable name is entered incorrectly, which generates an error message:

```
prompt> set_app_var sh_enabel_page_mode 1
Error: "sh_enabel_page_mode" is not an application variable
        Use error_info for more info. (CMD-013)
```

This example shows the variable name entered correctly:

```
prompt> set_app_var sh_enable_page_mode 1
1
```

This example resets the variable to its default value:

```
prompt> set_app_var sh_enable_page_mode -default
0
```

SEE ALSO

```
get_app_var(2)
report_app_var(2)
write_app_var(2)
```

set_case_analysis

Specifies that a port or pin is at a constant logic value 1, 0, or static, or is considered with a rising or falling transition.

SYNTAX

```
status set_case_analysis
  value
  port_or_pin_list
```

Data Types

value	string
port_or_pin_list	list

ARGUMENTS

value

Specifies a constant logic value or a transition to assign to the given pin or port. The valid constant values are **0**, **1**, **static\P, zero, and one. The valid transition values are rising, falling, rise, and fall.**

port_or_pin_list

Lists ports or pins to which the case analysis is assigned. In the case of leaf pins, constant propagation is only executed forward. No backward constant propagation is performed unless a hierarchical driver exists with a conflicting case value.

DESCRIPTION

Case analysis is a way of specifying a given mode for the design without altering the netlist structure. You can specify for the current timing analysis session, that some signals are at a constant value (**1**, **0**, or **static**), or that only one type of transition (**rising** or **falling**) is considered.

Pins of a design can be driven by logic values from the design, but if case analysis is used to set a value on such pins, the values set by case analysis dominate. If the case values are subsequently removed, the value driving the pin from the design are propagated.

When case analysis is specified as a constant value **0**, **1**, or **static**, this value is propagated through the network if the constant value is a controlling value for the traversed logic. For example, if you specify that one of the inputs of a NAND gate is a constant value of **0**, it is propagated to the NAND output, which is now considered at a logic constant of **1**. This propagated constant value is then propagated to all cells driven by this signal. Propagation of

logic **static** follows the same controlling logic convention. For example, if one input of an 2-input AND gate is **static** and the other is a logic **1**, then the output is **static**. However, if one input is **static** and the other input is **0**, then the output is **0**.

Applying **static** logic constant values in the design enables nets with **static** logic value signals will be filtered as acting aggressors in crosstalk analysis. Therefore, propagating **static** case values may be used to filter downstream paths from acting as aggressors without the side-effect of applying logic **1** or **0** constants in the design.

Note: When a logic value is propagated onto a net, the associated design rule checking (DRC_ constraint checks area disabled. However, the max_capacitance DRC check can be performed on driver pins for constant nets by setting **timing_enable_max_capacitance_set_case_analysis** variable to **true**. This variable is set to **false** by default.

In the event that the case analysis value is a transition, the given pin or port is considered only for timing analysis with the specified transition. The other transition is disabled. The case analysis information is used by all analysis commands.

Case analysis is used in addition to the mode commands to fully specify the mode of a design. For example, a design that instantiates models with a TESTMODE, is specified so that the TESTMODE is disabled during the timing analysis session by using the **set_mode** command. In addition, if a TESTMODE signal exists on the design, it is specified to a constant logic value, so that all test logic controlled by the TESTMODE signal is disabled.

The **set_case_analysis** command can be used for power analysis for both averaged mode and time-based mode in PrimeTime PX. Values set by the **set_case_analysis** command are also used for the internal logic simulation to get switching activities for nonannotated nets. However, the **-rising** and **-falling** options are ignored during power calculation. When simulation results from VCD or SAIF conflicts with the **set_case_analysis** command values, simulation wins.

EXAMPLES

The following example shows that the IN1 port is at a constant logic 0.

```
pt_shell> set_case_analysis 0 IN1
```

The following example shows that the IN1 port is at a constant logic static.

```
pt_shell> set_case_analysis static IN1
```

The following example shows that the pins U1/U2/A and U1/U3/CI are considered only for a rising transition. The falling transition on these pins is disabled.

```
pt_shell> set_case_analysis rising {U1/U2/A U1/U3/CI}
```

The following example specifies how to disable the TESTMODE of a design for which instances are models having a TESTMODE mode. The design also has a TEST_PORT port set to a constant logic value **0**.

```
pt_shell> remove_mode TESTMODE U1/U2
pt_shell> set_case_analysis 0 TEST_PORT
```

SEE ALSO

remove_case_analysis(2)

```
report_case_analysis(2)
report_timing(2)
set_case_sequential_propagation(2)
set_mode(2)
disable_case_analysis(3)
disable_case_analysis_ti_hi_lo(3)
```

set_clock_gating_check

Specifies the value of setup and hold time for clock gating checks.

SYNTAX

```
string set_clock_gating_check
   [-setup setup_value]
   [-hold hold_value]
   [-rise | -fall]
   [-high | -low]
   [object_list]
```

Data Types

```
setup_value     float
hold_value      float
object_list     list
```

ARGUMENTS

-setup *setup_value*

Specifies the clock gating setup time. The default is 0.0.

-hold *hold_value*

Specifies the clock gating hold time. The default is 0.0.

-rise

Specifies that only rising delays are constrained. By default, if neither the **-rise** or **-fall** options are specified, both rising and falling delays are constrained.

-fall

Specifies that only falling delays are constrained. By default, if neither the **-rise** or **-fall** options are specified, both rising and falling delays are constrained.

-high

Specifies that the check is performed on the high level of the clock. By default, PrimeTime determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates PrimeTime performs the check on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX and OR-AND) PrimeTime cannot determine which to use, and does not perform checks unless you specify either the **-high** or **-low** options. If the user-specified value differs from that derived by PrimeTime, the user-specified value takes precedence, and a warning message is issued. This option sets the attribute only on the specified pin or cell. If you specify the **-high** or **-low** options, you must also specify the *object_list*

variable; in that case, the *object_list* variable must not contain a clock.

-low

Indicates that the check is performed on the low level of the clock. By default, PrimeTime determines whether to use the high or low clock level using information from the cell's logic. That is, for AND and NAND gates, PrimeTime performs the check on the high level; for OR and NOR gates, on the low edge. For some complex cells (for example, MUX and OR-AND) PrimeTime cannot determine which to use, and does not perform checks unless you specify either the **-high** or **-low** options. If the user-specified value differs from that derived by PrimeTime, the user-specified value takes precedence, and a warning message is issued. This option sets the attribute only on the specified pin or cell. If you specify the **-high** or **-low** options, you must also specify the *object_list* variable; in that case, the *object_list* variable must not contain a clock.

object_list

Specifies a list of objects in the current design for which the clock gating check is applied. The objects can be clocks, pins, or cells. If a cell is specified, all input pins of that cell are affected. If a pin or cell is specified, any clock gating checks located at the specified objects are affected. If a clock is specified, the clock gating check is applied to all gating gates driven by that clock. If you specify the **-high** or **-low** options, you must also specify the *object_list* variable; in that case, the *object_list* variable must not contain a clock. By default, if the *object_list* variable is not specified, the clock gating check is applied to the current design.

DESCRIPTION

The **set_clock_gating_check** command specifies a setup or hold time clock gating check used for clocks, pins, or cells. The gating check is performed on pins that gate a clock signal.

The behavior of the **set_clock_gating_check** command is controlled by the **timing_clock_gating_check_fanout_compatibility** variable.

The clock gating setup check is used to ensure the controlling data signals are stable before the clock is active. This check is performed on combinational gates through which the clock signals are propagated. The arrival time of the leading edge of the clock pin is checked against both levels of any data signals gating the clock. A clock gating setup failure can cause either a glitch at the leading edge of the clock pulse, or a clipped clock pulse.

The clock gating hold check is used to ensure that the controlling data signals are stable while the clock is active. The arrival time of the trailing edge of the clock pin is checked against both levels of any data signal gating the clipped clock pulse.

The **-high** and **-low** options are intended to specify clock gating checks that PrimeTime cannot determine. These options apply only to the pins specified.

To remove information set by **set_clock_gating_check** command, use the **remove_clock_gating_check** command. The **reset_design** command removes all attributes from the design, including those set by the **set_clock_gating_check** command.

To report information for the clock gating check, use the **report_clock_gating_check** command.

Clock gating check constraints are applied in this order:

- Clock gating check set on specific pins or cells
- Clock gating constraint arcs from library or cells on which gating check is set
- Clock gating constraint set on clocks and design

EXAMPLES

The following example specifies a setup time of 0.2 and a hold time of 0.4 for all gates in the clock network of clock CK1.

```
pt_shell> set_clock_gating_check -setup 0.2 -hold 0.4 [get_clocks CK1]
```

The following example specifies a setup time of 0.5 on the and1 cell.

```
pt_shell> set_clock_gating_check -setup 0.5 [get_cells and1]
```

SEE ALSO

```
remove_clock_gating_check(2)
current_design(2)
report_constraint(2)
reset_design(2)
report_clock_gating_check(2)
timing_clock_gating_check_fanout_compatibility(3)
```

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis.

SYNTAX

```
status set_clock_groups
   -group clock_list
   [-exclusive]
   [-physically_exclusive]
   [-logically_exclusive]
   [-asynchronous]
   [-allow_paths]
   [-name name]
   [-comment comment_string]
```

Data Types

clock_list	list
name	string
comment_string	string

ARGUMENTS

-group *clock_list*

Specifies a list of clocks.

You can use this option multiple times in a single **set_clock_groups** command. Each **-group** iteration specifies a group of clocks, which are exclusive or asynchronous with the clocks in all other groups. If only one group is specified, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. Therefore, a default other group is created for this single group. Whenever a new clock is created, it is automatically included in the default exclusive or asynchronous group.

-exclusive

This is an alias for **-logically_exclusive** and is provided for backward compatibility.

-physically_exclusive

Specifies that the clock groups are physically exclusive with each other. Physically exclusive clocks cannot coexist in the design physically. An example of this is multiple clocks that are defined on the same source pin. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive

Logically exclusive clocks do not have any functional paths between them but might have coupling interactions with each other. An example of logically exclusive clocks is multiple clocks, which are selected by a MUX but can still interact through coupling upstream of the MUX cell. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous

Specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. Signal integrity analysis uses an infinite arrival window on the aggressor unless all the arrival windows on the victim net and the aggressor net are defined by synchronous clocks. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-allow_paths

Enable the timing analysis between specified asynchronous clock groups. The default behavior is to suppress timing paths between asynchronous clocks. The **-allow_paths** option must be used with **-asynchronous** option.

-name *name*

Specifies a name for the clock grouping to be created. Each command should specify a unique name, which identifies the exclusive or asynchronous relationship among specified clock groups, and this name is used later on to easily remove the clock grouping defined here. By default, the command creates a unique name. It takes the first clock name of the first list specified by -group option as the nucleus of the group name. if there is only one group list, then the unique string "_others" is appended. At last, the name is appended with a unique number, starting from 1.

-comment *comment_string*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis. This is similar to using the set_false_path command. In addition, these relationships also imply the type of crosstalk analysis which should be performed between the clocks. A clock cannot be present in multiple groups during one **set_clock_groups** command execution, but can be included in multiple groups by executing multiple **set_clock_groups** commands. Clock relationships are only implied across the specified clock groups; no relationship is implied across clocks within a group.

For all three relationships, timing paths between the clocks are suppressed. The relationships differ in how crosstalk is handled. If clocks are asynchronous, the clocks are assumed to have an infinite window relationship with each other. If clocks are physically exclusive, only one clock can act as an aggressor or victim during crosstalk analysis; interactions between the clocks are not explored. If clocks are logically exclusive, the crosstalk computation is computed normally (synchronously if the clocks are synchronous) even though the timing paths between the clocks are not reported.

Multiple relationship types can exist between two clocks. When this happens, the relationships obey the following precedence:

- Physically exclusive (highest)
- Asynchronous

- Logically exclusive (lowest)

These precedence rules reflect the real physical behavior of the resulting circuit. For example, if two clocks are asynchronous but they are never simultaneously present, then no crosstalk should be computed.

Note that the traditional **set_false_path** exception between clocks does not result in infinite window crosstalk computation between two clocks. If multiple clocks are asynchronous with each other, the asynchronous relationship should be specified with the **set_clock_groups** command. Failure to specify this relationship for asynchronous clocks could result in an optimistic analysis.

A special **-allow_paths** modifier can be used with the **-asynchronous** option. When used, the clocks are assumed to have an infinite window relationship for crosstalk purposes, but timing paths between the clocks are treated normally. When using this option, paths between the clock domains can be manually disabled using the **set_false_path** command.

When a clock pair is defined such that it has false paths (using either physically or logically exclusive or asynchronous) but subsequently modified to have **-asynchronous -allow_paths**, then this is an error and the first definition holds. In the reverse situation, also the redefinition to false paths from **-allow_paths** between the clock pairs would also be an error. See the UITE-457 and UITE-458 messages.

This command should be used to specify the relationships among clocks before using the **set_active_clocks** command. You should not also set a false path if you have already specified two clocks as exclusive or asynchronous. An error message is issued if you attempt to set a false path between two clocks which are already exclusive or asynchronous. If a false path was previously set between two clocks when an exclusive or asynchronous relationship is applied, the false path is overwritten by the **set_clock_groups** command. Other exceptions are unaffected.

A clock relationship on a master clock does not automatically propagate to any generated clocks. If the relationship should also apply to generated clocks, they should be explicitly included in the **set_clock_groups** command.

To undo the **set_clock_groups** command, use the **remove_clock_groups** command. To report the clock groups defined in a design, use the **report_clock** command with the **-groups** option.

EXAMPLES

The following example defines two asynchronous clock domains:

```
pt_shell> set_clock_groups -asynchronous -name g1 -group CLK33 -group CLK50
```

The following example defines a clock named *TESTCLK* to be asynchronous with all other clocks in the design:

```
pt_shell> set_clock_groups -asynchronous -group TESTCLK
```

The following example shows how to simultaneously analyze multiple clocks per register without manually specifying false paths. Assume two pairs of mutually exclusive clocks that are multiplexed:

- CLK1 and CLK2
- CLK3 and CLK4

If each pair of clocks is selected by a different signal, you must execute two **set_clock_groups** commands to specify two independent exclusivity relationships, one for each MUX selection signal:

```
pt_shell> set_clock_groups -physically_exclusive -group CLK1 -group CLK2
pt_shell> set_clock_groups -physically_exclusive -group CLK3 -group CLK4
```

If each pair of clocks is selected by a single MUX selection signal, you would execute only one **set_clock_groups** command to simultaneously analyze all four clocks.

```
pt_shell> set_clock_groups -physically_exclusive -group {CLK1 CLK3} -group {CLK2 CLK4}
```

This example does not imply any type of relationship between CLK1-CLK3 or CLK2-CLK4. For example, f CLK1 and CLK3 were also asynchronous with each other, you would need to specify an additional relationship between them.

SEE ALSO

```
create_clock(2)
create_generated_clock(2)
remove_clock_groups(2)
report_clock(2)
set_active_clocks(2)
set_false_path(2)
```

set_clock_latency

Specifies the latency of the clock network.

SYNTAX

```
status set_clock_latency
   [-clock clock_list]
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-source]
   [-dynamic dynamic_component_of_delay]
   [-late]
   [-early]
   [-pll_shift]
   [-generated]
   [-primary]
   [-clock_fall]
   [-clock_rise]
   delay
   object_list
```

Data Types

clock_list	list
dynamic_component_of_delay	float
delay	float
object_list	list

ARGUMENTS

-clock *clock_list*

Specifies a list of clock objects associated with the network latency placed on all pin/port objects in the object list. If the **-clock** option is used when object list refers to clock objects, a warning is issued that the option is not relevant in this case and execution of the command proceeds as if the **-clock** option was not given. A more specific clock latency setting overrides a more general one.

-rise

Specifies clock rise latency.

-fall

Specifies clock fall latency.

-min

Specifies clock latency for the minimum operating condition.

-max

Specifies clock latency for the maximum operating condition.

-source

Specifies clock source latency.

-dynamic *dynamic_component_of_delay*

Specifies dynamic component of clock latency value.

-late

Specifies clock late source latency.

-early

Specifies clock early source latency.

-pll_shift

Specifies the phase error for output clocks coming out of phase locked loops (PLLs). This option adds the specified delay to the phase adjustment of the PLL.

-generated

Specifies generated clock signal type. Proper sense should be set on the clock definition point by **set_sense** before applying this option. Please note that this option is only valid for clock definition points.

-primary

Specifies primary clock signal type (default). Proper sense should be set on the clock definition point by **set_sense** before applying this option. Please note that this option is only valid for clock definition points.

-clock_fall

Specifies that the signal is coming from the falling edge of the clock. If not specified, it uses the falling edge for **-fall**. Proper sense should be set on the clock definition point by **set_sense** before applying this option. Please note that this option is only valid for clock definition points.

-clock_rise

Specifies that the signal is coming from the rising edge of the clock. If not specified, it uses the rising edge for **-rise**. Proper sense should be set on the clock definition point by **set_sense** before applying this option. Please note that this option is only valid for clock definition points.

delay

Specifies clock latency value.

object_list

Specifies a list of clocks, pins, or ports.

DESCRIPTION

Two types of clock latency can be specified for a design:

- Clock network latency - This is the time it takes a clock signal to propagate from the clock definition point to a register clock pin. The rise and fall latencies are the latencies for rising and falling transitions at the register clock pin, respectively. Inversion of the clock waveform, if present in the clock network, is not taken into consideration when computing clock network latencies at register clock pins.
- Clock source latency - This is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. It can be used to model off-chip clock latency when the clock generation circuit is not part of the current design. You can use clock source latency for generated clocks to model the delay from master-clock to generated-clock definition point.

Dynamic clock latency is the component of the total clock latency which is considered dynamic in nature. A dynamic value might differ along successive clock cycles and therefore can only be used to calculate CRP if a zero-cycle path is being considered. A zero-cycle path is one in which the same clock edge both launches and captures.

To specify the dynamic component of any clock latency, use the **-dynamic** option. It can only be specified if the **-source** option and total clock latency is also specified with the command. You can specify dynamic latency on only clock objects, or pin or port objects that are clock sources. When a pin or port object is specified, the clock also has to be specified by **-clock** option.

PrimeTime assumes ideal clocking, which means clocks have a specified network latency, designated by the **set_clock_latency** command, or zero network latency, by default. Propagated clock network latency, designated by the **set_propagated_clock** command, is normally used for post-layout after final clock tree generation. Ideal clock network latency provides an estimate of the clock tree for pre-layout.

You can specify clock source latency for ideal or propagated clocks. The total clock latency at a register clock pin is the sum of clock source latency and clock network latency. If the rise and fall latencies are different, then the source latencies for a latch are picked based on the sense at the clock port and network latencies are picked based on the sense at the latch clock pin. Thus, source latency takes into account the clock inversions on the clock network, but network latency does not.

In bc_wc analysis mode, the **-min** and **-max** options specify the corner for the clock latency value (fast or slow), and the **-early** and **-late** options specify the early or late latency value within the corner. In the on_chip_analysis mode, the min/max operating conditions are the variation bounds within the single corner analysis. Therefore, only the min-early and max-late latencies are used for the analysis. In the on_chip_variation mode, it is sufficient to specify either the **-early** or **-late** options or the **-min** or **-max** options for the **set_clock_latency** command.

For internally-generated clocks, PrimeTime automatically computes the clock source latency provided that the master clock of the generated clock has propagated latency and no user-specified value for the generated clock source latency exists. If the master clock is ideal, the master clock has source latency, and there is no user-specified value for the generated clock's source latency, then zero source latency is assumed.

If the **set_clock_latency** command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. Directly setting a network latency makes the affected registers ideal. A warning is issued if the **set_propagated_clock** command has previously set a propagated attribute on the same object (clock, pin, or port). Clock source latencies are allowed only on clocks and clock source pins.

No check is performed at the UI level that a clock specified with the **-clock** option passes through the pin or pins referenced in the object list or not. If the clock specified does not pass through the pin, the command has no effect but no warning of this fact is given.

To undo the **set_clock_latency** command, use the **remove_clock_latency** command.

To see clock network and source latency information (including dynamic component), use the **report_clock** command with the **-skew** option.

EXAMPLES

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for the CLK1 clock.

```
pt_shell> set_clock_latency 1.2 \
          -rise [get_clocks CLK1]
pt_shell> set_clock_latency 0.9 \
          -fall [get_clocks CLK1]
```

The following example specifies an early rise and a fall source latency of 0.8 and a late rise and fall source latency of 0.9 for the CLK1 clock.

```
pt_shell> set_clock_latency 0.8 -source \
          -early [get_clocks CLK1]
pt_shell> set_clock_latency 0.9 -source \
          -late  [get_clocks CLK1]
```

The following example specifies an early and late source latency of 3 and 5, respectively, with dynamic components of 0.5.

```
pt_shell> set_clock_latency 3 -source \
          -early -dynamic 0.5  [get_clocks CLK1]
pt_shell> set_clock_latency 5 -source \
          -late  -dynamic 0.5 [get_clocks CLK1]
```

The following example specifies the source latency as well as the dynamic portion of source latency on a clock source pin.

```
pt_shell> create_clock -name clk -period 10 [gte_pins clk_source_pin]
pt_shell> set_clock_latency 100 -source [get_pins clk_source_pin] \
          -dynamic 20 -clock [get_clocks CLK1] -source
```

The following example specifies generated clock source latencies of 0.5 to the rise-to-rise clock edge and 0.6 to the fall-to-fall clock edge on pin and2/Y.

```
pt_shell> set_clock_latency 0.5 -source \
          -generated -rise -clock_rise and2/Y
pt_shell> set_clock_latency 0.6 -source \
          -generated -fall -clock_fall and2/Y
```

The following example specifies primary clock source latencies of 0.1 to the rise-to-rise clock edge and 0.2 to the fall-to-fall clock edge on pin and2/Y.

```
pt_shell> set_clock_latency 0.1 -source \
          -primary -rise -clock_rise and2/Y
pt_shell> set_clock_latency 0.2 -source \
          -primary -fall -clock_fall and2/Y
```

The following example shows a timing report where dynamic clock latency has been specified with the **set_clock_latency** command:

```
pt_shell> report_timing -path_type full_clock_expanded
****************************************
Report : timing
        -path_type full_clock_expanded
        -delay_type max
        -max_paths 1
Design : latency
 ...
****************************************

  Startpoint: ff1 (rising edge-triggered flip-flop clocked by clk)
```

```
Endpoint: ff2 (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock source latency                     3.00       3.00
clock source latency (dynamic)           0.60       3.60
clk (in)                                 0.00       3.60 r
u1/Z (IBUF1)                             0.85       4.45 r
ff1/CP (FD1)                             0.00       4.45 r
ff1/Q (FD1)                              1.44       5.89 f
u3/Z (IV)                                0.58       6.47 r
ff2/D (FD1)                              0.00       6.47 r
data arrival time                                   6.47

clock clk (rise edge)                   10.00      10.00
clock source latency                     1.00      11.00
clock source latency (dynamic)          -0.50      10.50
clk (in)                                 0.00      10.50 r
u1/Z (IBUF1)                             0.85      11.35 r
u2/Z (IBUF1)                             0.68      12.03 r
ff2/CP (FD1)                             0.00      12.03 r
clock reconvergence pessimism            2.00      14.03
library setup time                      -0.80      13.23
data required time                                 13.23
-------------------------------------------------------------
data required time                                 13.23
data arrival time                                  -6.47
-------------------------------------------------------------
slack (MET)                                         6.76
```

SEE ALSO

remove_clock_latency(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_propagated_clock(2)

set_clock_map

Defines the mapping of clocks across hierarchical design levels for HyperScale analysis, overriding the default inferred clock mapping.

SYNTAX

```
status set_clock_map
   [-parent_clock parent_clock_name]
   [-block_clocks block_clock_names]
   [-instances inst_cells]
   [-add]
   clock_list
```

Data Types

parent_clock_name	string
block_clock_names	list
inst_cells	list
clock_list	list

ARGUMENTS

-parent_clock *parent_clock_name*

Specifies the name of the clock used at the parent hierarchy level that is the same as the clocks listed in the *clock_list*. This option is valid only for block-level analysis and cannot be used with the **-block_clocks** option.

-block_clocks *block_clock_names*

Specifies the names of the clocks used at the subblock level that are the same as the clock identified in the *clock_list*. This option is valid only for analysis at a higher level of hierarchy and cannot be used with the **-parent_clock** option.

-instances *inst_cells*

Specifies a list of hierarchical subblock instances in the current design for which the clock mapping applies. Clock mapping can be instance-specific even when different instances refer to same physical block. This option can be used only with the **-block_clocks** option.

-add

Adds this clock mapping to the default inferred clock associations. Without this option, you take full control of the clock mapping to the parent level or block level; the new clock mapping overwrites any existing clock mappings inferred by the tool. This option does not affect clock mappings defined by previous **set_clock_map** commands; such mappings remain in effect until you use the **remove_clock_map** command.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names in the current design. The command maps these clocks to the parent clock named by the **-parent_clock** option or to the lower-level clocks named by the **-block_clocks** option.

DESCRIPTION

In a hierarchical design flow, top-level and block-level designs often need to be implemented and analyzed separately, with separate but highly associated block-level and top-level timing constraints. The same clocks might be assigned different names at different hierarchical levels.

The PrimeTime tool automatically detects and associates clocks with mismatching names across hierarchical levels by considering the clock characteristics. In the HyperScale flow, the tool usually establishes the clock mapping correctly, especially when you use HyperScale constraint extraction.

However, in complex or ambiguous clocking situations with multiple clocks, using the available design information, the tool cannot always associate these clocks as intended. To explicitly specify the mapping and override the default inferred mapping, use the **set_clock_map** command with the appropriate options:

- To define the mapping of clocks at the current level with clocks at the parent level, use the **-parent_clock** option.
- To define the mapping of clocks at the current level with clocks at the subblock level, use the **-block_clocks** option, and optionally use the **-instances** option to apply the mapping to specific instances.

You cannot use the **-parent_clock** and **-block_clocks** options at the same time. To define both parent and subblock mappings for an intermediate-level design, use two separate **set_clock_map** commands.

To remove clock mappings set with the **set_clock_map** command, use the **remove_clock_map** command.

Reporting the Clock Mapping

To show the clock mappings used by the tool for timing analysis, use the **report_clock -map** command. The report shows an asterisk next to each clock mapped explicitly by the **report_clock -map** command:

```
pt_shell> set_clock_map MCLK -block_clocks SCLK
Information: Abandoning fast timing updates. (PTE-018)
1
pt_shell> update_timing -full
Information: Loading block timing data for block instance low_1. (HS-001)
Information: applying user mapping between block clock 'SCLK' and clock 'MCLK'.
Information: Loading block timing data for block instance low_1. (HS-001)
1
pt_shell> report_clock -map
 ...
Instance         Top level clock     Block level clock
-------------------------------------------------------------
low_1            (N/A)               MCLK
                 MCLK                SCLK*
```

Situations That Require Explicit Clock Mapping

The most common type of ambiguity is caused by clocks with same period and waveform propagating though overlapping networks across a block boundary. For example, multiple top-level clocks with same basic period and waveform characteristics can all reach a single block port and enter into a block. In this case, block-level constraints have multiple clocks defined on the port to represent all the top clocks. Explicitly defining the mapping with the **set_clock_map** command is the most reliable way to ensure correct mapping.

Another common ambiguity is caused by virtual clocks, which are used to model a clock signal without a physical source (and network) present in the current level of design. For example, even when a top-level clock does not physically propagate into a specific hierarchical block itself, it can still launch data signals that enter the block or capture signals that leave the block. To model such a clock in block-level constraints, you can define a virtual clock with a period and waveform matching the top-level physical clock, but without attaching it to any source or source network objects in the block design. Ambiguity arises when multiple top-level clocks have the same characteristics. Use the **set_clock_map** command to resolve such ambiguities.

You can choose to overdesign at the block level by defining clocks with tighter margins or higher frequencies than the same clocks at the top level. When the clock characteristics do not match exactly between the block-level and top-level clocks, the tool does not establish mapping between the clocks by default because of the clock differences. To override this default behavior, use the **set_clock_map** command to explicitly map the slightly mismatching clocks. In this situation, the **report_constraint** command still reports the mismatching clocks as a boundary constraint violations.

You can modify the behavior of automatic clock mapping by setting the variables **hier_disable_auto_clock_mapping** and **hier_clock_mapping_period_percent_tolerance**.

Clock Mapping and HyperScale Technology

Use the **set_clock_map** command in a manner consistent with the HyperScale configuration command **set_hier_config**. For example, for block-level analysis, a clock is defined with the name BCLK, which represents a top-level counterpart with the name TCLK. (Note that TCLK is defined in a separate set of top-level constraints and is not included in the constraints for block-level analysis.) Use the following commands at the block level:

```
set_hier_config -parent -path $top_dir ...
 ...
create_clock -name BCLK ...
 ...
set_clock_map BCLK -parent_clock TCLK
 ...
update_timing
 ...
```

For top-level analysis, the clock is defined with name TCLK:

```
set_hier_config -block A_BLK -path $blk_dir ...
 ...
create_clock -name TCLK ...
 ...
set_clock_map TCLK -block_clocks BCLK
 ...
update_timing
 ...
```

Mapping a Top-Level Clock to Multiple Block-Level Clocks

HyperScale technology supports the mapping of one top-level clock to multiple block-level clocks. However, in the opposite direction, a block-level clock maps to exactly one top-level clock. Most often, with a top-level clock propagating into a block through a single port, a single block-level clock is defined on the port and uniquely maps to the top-level clock. However, there are cases a single block-level clock definition cannot fully represent the top-level clock intent. For example:

- Multiple waveforms from the same clock -- The top-level clock enters into a block through multiple ports with a positive waveform on some ports but inverted through others. This requires block-level constraints to define two clocks with opposite waveforms on the associated ports.
- Re-entrant generated clocks -- This happens when a generated clock defined inside a block propagates outside of the block and re-enters the same block through some port. A generated clock requires the

source network path back to its master clock to have a valid source latency computation. The portion of the source network traveling outside of the block does not exist for block-level analysis, so block-level constraints need to define two separate clocks to fully represent the single generated clock in top-level constraints.

By default, the tool automatically establishes 1-to-N clock mapping for most of the these configurations. However, if the design is too complex or ambiguous for reliable automatic mapping, you should use the **set_clock_map** command to explicitly specify the mapping.

EXAMPLES

The following example defines the mapping of top-level clock SYSCLK to block-level clock BCLK in HyperScale top-level analysis.

```
# Map SYSCLK at current level to BCLK in HyperScale block
set_clock_map SYSCLK -block_clocks BCLK
```

The following example defines the mapping of top-level clock SYSCLK to block-level clock BCLK_1 for one instance and BCLK_2 for another instance.

```
set_clock_map SYSCLK -block_clocks BCLK_1 -instance blk_inst1
set_clock_map SYSCLK -block_clocks BCLK_2 -instance blk_inst2
```

In the following block-level analysis example, two block-level clocks map to the same top-level clock.

```
set_clock_map sysCLK_pos -parent_clock SYSCLK
set_clock_map sysCLK_neg -parent_clock SYSCLK
```

The following example shows the clock mapping reported with both automatically resolved and user-specified explicit clock mappings. User-specified mappings are annotated with an asterisk (*) in the report.

```
pt_shell> report_clock -map
 ...
Instance       Top level clock     Block level clock
-----------------------------------------------------
core3          CLK1                CLK1* CLK1_v*
               CLK2                CLK2  CLK2_v
               CLK3                CLK3  CLK3_v
               CLK4                CLK4* CLK4_v*
core4          CLK1                CLK1* CLK1_v*
               CLK2                CLK2  CLK2_v
               CLK3                CLK3  CLK3_v
               CLK4                CLK4  CLK4_v
core1          CLK1                CLK1* CLK1_v*
               CLK2                CLK2* CLK2_v*
               CLK3                CLK3  CLK3_v
               CLK4                CLK4  CLK4_v
core2          CLK1                CLK1* CLK1_v*
               CLK2                CLK2* CLK2_v*
               CLK3                CLK3  CLK3_v
               CLK4                CLK4  CLK4_v
```

SEE ALSO

create_clock(2)

```
create_generated_clock(2)
remove_clock_map(2)
report_clock(2)
report_constraint(2)
set_hier_config(2)
clock_mapping_violations(3)
hier_clock_mapping_period_percent_tolerance(3)
hier_disable_auto_clock_mapping(3)
```

set_clock_sense

The **set_clock_sense** command is deprecated. Use the **set_sense** command instead.

SEE ALSO

`remove_sense(2)`
`set_sense(2)`

set_clock_transition

Specifies transition time of register clock pins.

SYNTAX

```
string set_clock_transition
   [-rise]
   [-fall]
   [-min]
   [-max]
   transition
   clock_list
```

Data Types

transition	float
clock_list	list

ARGUMENTS

-rise

Specifies clock transition time for rising clock edge.

-fall

Specifies clock transition time for falling clock edge.

-min

Specifies clock transition time for minimum conditions.

-max

Specifies clock transition time for maximum conditions.

transition

Specifies transition time of clock pins.

clock_list

Provides a list of clocks in the design. The transition times on all register clock pins and clock gating constraint arcs in the transitive fanout of specified clock are affected. You only can specify clocks with ideal latency in the list. When affected clock pins in the fanout of a clock are marked with propagated latency, the values are ignored in the **set_clock_transition** command.

DESCRIPTION

This command provides the ability to override the calculated transition times on clock pins of registers and clock gating constraint arcs.

This command is useful for pre-layout when clock trees are incomplete and using calculated transition times at register clock pins and for clock gating constraint arcs can be highly pessimistic. The transition value specified with this command overrides the transition times of all nets directly feeding a sequential element clocked by the specified clock.

Use the **set_clock_transition** command only with ideal clocks. For propagated clocks the calculated transition times are used. Propagated clocks are only set after the final clock tree is constructed.

If a clock transition is not specified for an ideal clock, and if the timing_ideal_clock_zero_default_transition variable is TRUE (by default it is TRUE), then zero transition is used. Otherwise, the transition time is calculated as it is for other pins in the design.

To undo the **set_clock_transition** command, use the **remove_clock_transition** command.

To list all clock transition values which have been set, use **report_clock -skew**.

EXAMPLES

This example specifies a rise transition time of 0.38 and fall transition time of 0.25 for register clock pins clocked by "CLK1".

```
pt_shell> set_clock_transition 0.38 -rise [get_clocks CLK1]
pt_shell> set_clock_transition 0.25 -fall [get_clocks CLK1]
```

SEE ALSO

```
remove_clock_transition(2)
report_clock(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_propagated_clock(2)
timing_ideal_clock_zero_default_transition(3)
```

set_clock_uncertainty

Specifies the uncertainty (skew) of specified clock networks.

SYNTAX

```
string set_clock_uncertainty
   [-rise]
   [-fall]
   [-setup]
   [-hold]
   uncertainty
   [object_list
    -from from_clock
      | -rise_from rise_from_clock
      | -fall_from fall_from_clock
    -to to_clock
      | -rise_to rise_to_clock
      | -fall_to fall_to_clock]
```

Data Types

```
object_list          list
from_clock           list
rise_from_clock      list
fall_from_clock      list
rise_to_clock        list
fall_to_clock        list
to_clock             list
uncertainty          float
```

ARGUMENTS

-rise

Indicates that the *uncertainty* option applies to only the rising edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-rise_to* option instead.

-fall

Indicates that the *uncertainty* option applies to only the falling edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-fall_to* option instead.

-setup

Indicates that the *uncertainty* option applies only to setup checks. By default, the *uncertainty* option applies to both setup and hold checks.

-hold

Indicates that the *uncertainty* option applies only to hold checks. By default, the *uncertainty* option applies to both setup and hold checks.

uncertainty

A floating point number that specifies the uncertainty value. Typically, clock uncertainty should be positive. Negative uncertainty values are supported for constraining designs with complex clock relationships. Setting the uncertainty value to a negative number could lead to optimistic timing analysis and should be used with extreme care.

-from *from_clock*

Specifies the source clock for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

-rise_from *rise_from_clock*

This is the same as the *-from* option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options.

-fall_from *fall_from_clock*

This is the same as the *-from* option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options.

-to *to_clock*

Specifies the destination clock for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

-rise_to *rise_to_clock*

This is the same as the *-to* option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options.

-fall_to *fall_to_clock*

This is the same as the *-to* option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options.

object_list

Specifies a list of clocks, ports, or pins for simple uncertainty; the uncertainty is applied either to capturing latches clocked by one of the clocks in the *object list* option, or capturing latches whose clock pins are in the fanout of a port or pin specified in the *object_list* option. You must specify either the pair of the *-from* and *-to*, or *object_list* options; you cannot specify both.

DESCRIPTION

Specifies the clock uncertainty (skew characteristics) of specified clock networks. This command can specify either interclock uncertainty or simple uncertainty. For interclock uncertainty, use the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to* options to specify the source clock and the destination clock; all paths between these receive the uncertainty value. For simple uncertainty, use the *object_list* option; the uncertainty value is either to capturing latches clocked by one of the clocks in the *object list* option, or capturing latches whose clock pins are in the fanout of a port or pin specified in the *object_list* option.

Set the uncertainty to the worst skew expected to the endpoint or between the clock domains. You can increase the value to account for additional margin for setup and hold.

For crosstalk analysis, clock uncertainty doesn't change the arrival windows used in overlap analysis. However, the presence of clock uncertainty could change the borrow behavior in level-sensitive latch designs. Level sensitive latches may start borrowing in the presence of clock uncertainty leading to change in delta delays for logic nets that are in the fanout of affected latches.

When you specify interclock uncertainty, ensure that you specify it for all possible interactions of clock domains. For example, if you specify paths from CLKA to CLKB and CLKB to CLKA you must specify the uncertainty for both even if the values are the same. See the Examples section.

Interclock uncertainty is more specific than simple uncertainty. If a command that specifies interclock uncertainty conflicts with a command that specifies simple uncertainty, the command that specifies interclock uncertainty takes precedence. See the Examples section.

If there is no applicable interclock uncertainty for a path, the value for simple uncertainty is used. See the Examples section.

To remove the uncertainties set by the **set_clock_uncertainty** command, use the **remove_clock_uncertainty** command.

To view clock uncertainty information, use the **report_clock -skew** command.

EXAMPLES

The following example specifies that all paths leading to registers or ports clocked by CLK have setup uncertainty of 0.65 and hold uncertainty of 0.45.

```
pt_shell> set_clock_uncertainty -setup 0.65 [get_clocks CLK]
pt_shell> set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains.

```
pt_shell> set_clock_uncertainty 0.4 -from PHI1 -to PHI1
pt_shell> set_clock_uncertainty 0.4 -from PHI2 -to PHI2
pt_shell> set_clock_uncertainty 1.1 -from PHI1 -to PHI2
pt_shell> set_clock_uncertainty 1.1 -from PHI2 -to PHI1
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains with specific edges.

```
pt_shell> set_clock_uncertainty 0.4 -rise_from PHI1 -to PHI2
pt_shell> set_clock_uncertainty 0.4 -fall_from PHI2 -rise_to PHI2
pt_shell> set_clock_uncertainty 1.1 -from PHI1 -fall_to PHI2
```

The following example shows conflicts in the **set_clock_uncertainty** commands, one for simple uncertainty and one for interclock uncertainty. The interclock uncertainty value of 2 takes precedence.

```
pt_shell> set_clock_uncertainty 5 [get_clocks CLKA]
pt_shell> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example specifies the uncertainty from CLKA to CLKB and from CLKB to CLKA. Notice that both must be specified even though the value is the same for both.

```
pt_shell> set_clock_uncertainty 2 -from [get_clocks CLKA] -to [get_clocks CLKB]
pt_shell> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example illustrates a situation in which simple uncertainty is used when there is no applicable interclock uncertainty for a path. The first command specifies a simple uncertainty of 5 for CLKA paths, and the second command specifies an interclock uncertainty of 2 for paths from CLKB to CLKA. If there are paths between CLKA and other clocks (for example, CLKC, CLKD, ...) for which interclock uncertainty has not been specifically defined, the simple uncertainty (in this case, 5) is used.

```
pt_shell> set_clock_uncertainty 5 [get_clocks CLKA]
pt_shell> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

SEE ALSO

remove_clock_uncertainty(2)
report_clock(2)
set_clock_latency(2)
set_clock_transition(2)
timing_propagate_interclock_uncertainty(3)

set_connection_class

Sets the connection class value on ports.

SYNTAX

```
int set_connection_class
   connection_class_value
   object_list
```

Data Types

connection_class_value	string
object_list	list

ARGUMENTS

connection_class_value

Specifies the desired *connection_class* value of ports contained in the *object_list* option. The *connection_class_value* option is a single string value. This string can contain any number of space separated connection classes (see the example below).

object_list

Specifies ports whose connection classes are set.

DESCRIPTION

The *connection class* label is an attribute that is used to describe connection requirements for a given technology. Only those loads and drivers with the same connection class label can be legally connected. The labels "universal" and "default" are reserved labels. The "universal" label indicates that a pin or port can legally connect with any other load or driver. The "default" label is used for those library pins that do not otherwise have a connection class assigned to them through any library attributes. For designs being optimized analyzed, the "universal" label is assigned to those ports that do not otherwise have a connection class assigned to them. To change the default connection class label for those ports, use the *default_port_connection_class* variable.

Connection classes are placed on ports of designs in a technology library by using Library Compiler. The **set_connection_class** command places connection classes on ports of designs being analyzed (such as the current design). This is used to indicate a connection class requirement for the network that is connected to the specified port.

To view connection class values on ports, use the **get_attribute** command. To check your design for connection

class violations, use the **report_constraint** command.

EXAMPLES

The following example sets a connection class "internal" to the port named "xyz" in the current design.

```
pt_shell> set_connection_class  internal xyz
```

The following example sets the connection classes "internal" and "external" on the port "out" in the design "test".

```
dc_shell> set_connection_class "internal
                        external" test/out
```

In the following example, the connection class on a port is removed.

```
pt_shell> remove_connection_class  test/out
```

SEE ALSO

```
remove_connection_class(2)
report_constraint(2)
```

set_context_margin

Specifies the margin by which to tighten or relax constraints.

SYNTAX

```
status set_context_margin
   [-transition]
   [-delay]
   [-noise]
   [-max]
   [-min]
   [-rise]
   [-fall]
   [-above]
   [-below]
   [-high]
   [-low]
   [-percent]
   [-relax]
   [-height]
   [-width]
   [-objects cell_block_name]
   [-clock_only]
   [-data_only]
   margin_value
```

Data Types

cell_block_name	string
margin_value	double

ARGUMENTS

-transition

Specifies the margin for transition context at block boundary.

-delay

Specifies the margin for delay context at block boundary.

-noise

Specifies the margin for noise related context at block boundary.

-max

Specifies the margin for max context.

-min

Specifies the margin for min context.

-rise

Specifies the margin for rise transition or delay context at block boundary.

-fall

Specifies the margin for fall transition or delay context at block boundary.

-above

Specifies the margin for noise above voltage level context at block boundary.

-below

Specifies the margin for noise below voltage level context at block boundary.

-high

Specifies the margin for noise supply voltage level context at block boundary.

-low

Specifies the margin for noise ground voltage level context at block boundary.

-percent

Specifies the margin value as a percentage of delay.

-relax

Specifies the margin value as relax instead of tightening.

-height

Specifies the margin for noise bump height context.

-width

Specifies the margin for noise bump width context.

-objects *cell_block_name*

Specifies a list of instances or blocks for which the context is applied with margin.

-clock_only

Specifies whether the margin only to be applied to clock paths, this option cannot be used together with "-data_only" .

-data_only

Specifies whether the margin only to be applied to data paths, this option cannot be used together with "-clock_only" .

margin_value

Determines the margin value to be applied for HyperScale context.

DESCRIPTION

The **set_context_margin** command is used to specify the margin to be applied to the context automatically generated during analysis for the specified objects. The margin can be specified as an absolute value or as a percentage of the constraint. If no "-objects" is specified, the margin will apply to all the subblocks.

By default, the input and output delays are adjusted such that they are more constraining; that is, the specified margin is added to the max delay values and subtracted from the min delay values. If you specify the *-relax* option, the margin is subtracted from the max delay values and added to the min delay values.

The margin applies to the current design if no objects is specified. If no margin is specified, the default is 0.0.

By default, the input and output boundary context for a subblock is captured accurately with the values computed during timing or noise analysis. This command adds margin to the context to help improve the convergence between block and top-level timing analysis. Among the three options: **-transition**, **-delay**, and **-noise**, only one of them can be specified per command. If none of them is specified, "-transition" will be used as default. If neither "-clock_only" nor "-data_only" is specified, margin will be applied to both clock paths and data paths.

The margin value is interpreted as absolute value and the units are the applicable time or voltage units from the logic library currently used.

EXAMPLES

The following command specifies that a margin 0.5 should be added to all max values of input and output delays constraints and subtracted from all min values of input and output delay constraints. The margin applies to all objects in the current design.

```
pt_shell> set_context_margin 0.5 -delay
```

The following command specifies that the maximum value input delays on I2 should be reduced by 10 percent.

```
pt_shell> set_context_margin -max -delay -percent 10.0 -objects I2
```

SEE ALSO

characterize_context(2)
report_context(2)
write_context(2)

set_coupling_separation

Creates a separation constraint on nets.

SYNTAX

```
status set_coupling_separation
   [-pairwise pnets]
   nets
```

Data Types

pnets	list
nets	list

ARGUMENTS

-pairwise *pnets*

Specifies that coupling capacitances between only *pnets* and *nets* are excluded. This is referred to as "pairwise exclusion.

nets

Specifies the nets for which the separation constraint is applied. If you do not use the **-pairwise** option, the separation constraint is applied for this net with respect to all its aggressors.

DESCRIPTION

By default, all nets with coupling capacitors (non-filtered) are included in crosstalk and noise analysis as both victim and aggressor. This command creates a separation constraint on nets which means that all coupling capacitors on each net in *nets* are excluded for noise and crosstalk analysis.

From an analysis point of view, this command has the same effect of applying both the **set_si_delay_analysis** and **set_si_noise_analysis** commands with the **-exclude** option and specifying *nets* as both victims and aggressors. However, this constraint takes precedence over the constraints set by the **set_si_delay_analysis** and **set_si_noise_analysis** command.

Instances of this command are recorded for output by the **write_changes** command. This feature allows you to communicate the separation constraint to other implementation tools along with netlist changes.

To remove the result of this command, use the **remove_coupling_separation** command.

To view the result of this command, use the **report_si_delay_analysis** or **report_si_noise_analysis**

command with the **-coupling_separated** option.

EXAMPLES

The following example sets a separation constraint on all nets referred to by *CLK_NET_*.

```
pt_shell> set_coupling_separation [get_nets CLK_NET*]
1
```

The following example sets a separation constraint between net n3 and n1 and n2. It does not affect analysis of cross-coupling between n1 and n2 nor n3 and any other net.

```
pt_shell> set_coupling_separation -pairwise [get_nets {n1 n2}] [get_nets n3]
1
```

SEE ALSO

```
remove_coupling_separation(2)
report_si_delay_analysis(2)
report_si_noise_analysis(2)
set_si_delay_analysis(2)
set_si_noise_analysis(2)
write_changes(2)
```

set_cross_voltage_domain_analysis_guardband

Sets incremental derating factors to be used during cross-voltage-domain analysis for specified UPF supply nets or library cells.

SYNTAX

```
status set_cross_voltage_domain_analysis_guardband
   -early | -late
   -cell_check
   derate_value
   object_list
```

Data Types

```
derate_value        float
object_list         list
```

ARGUMENTS

-early

Applies the *derate_value* to early delays (shortest paths) during cross-voltage-domain analysis.

-late

Applies the *derate_value* to late delays (longest paths) during cross-voltage-domain analysis. The **-early** and **-late** options are mutually exclusive.

-cell_check

Applies the *derate_value* to cell_check arcs as with set_timing_derate the value specified combined with -late is used for setup checks and the value specified combined with -early is used for hold checks. This option is intended to derate the lauch capture delay difference inside of extracted timing models. To achieve this in pessimistic fashion it apllies the guardband derate by simple multiplication even to negative check values.

derate_value

Specifies the timing derate value applied as a scalar multiplier to cell delays during cross-voltage-domain analysis. This factor is applied in addition to any derate value specified with the **set_timing_derate** command or any advanced on-chip variation analysis invoked with the **timing_aocvm_enable_analysis** variable.

object_list

Specifies a collection of either UPF supply nets or library cells. If one or more supply nets are specified, the derating adjustment applies to all cells connected to those supply nets. If one or more library cells are specified, the derating adjustment applies to all instances of those cells. If the *object_list* argument is not used, the guard band setting applies to all UPF supply nets in the design.

DESCRIPTION

Cross-voltage-domain analysis uses two separate means of modeling the timing behavior of cells due to different supply voltage values. The preferred approach is to use the PrimeTime multivoltage delay calculation capability by providing libraries characterized at the relevant corner voltages using the **define_scaling_lib_group** command. If library characterization information at different voltage values is not available, the **set_cross_voltage_domain_analysis_guardband** command can be used to model the change in timing behavior of cells at different voltage values by derating the calculated delays.

If no object list is provided in the command, the specified *derate_value* applies to all UPF supply nets in the design. If an object list is provided, it can be heterogeneous collection of either UPF supply nets or library cells.

Derating adjustment values can be set for both a UPF supply net collection and a library cell collection. Both adjustments are applied where applicable, one in addition to the other. Any new guard band set for a supply net collection overwrites any previous settings for supply nets. Similarly, any new guard band set for a library cell collection overwrites any previous settings for library cells.

The **set_cross_voltage_domain_analysis_guardband** derate is an incremental derate applied in addition to any other derate specified through the **set_timing_derate** command or advanced on-chip variation (AOCV) analysis. It is not recommended to use **set_cross_voltage_domain_analysis_guardband** in addition to **define_scaling_lib_group**. However, if you use both commands, the timing value obtained from the scaling lib group analysis is also derated according to the *derate_value* specified by the **set_cross_voltage_domain_analysis_guardband** command.

EXAMPLES

The following example sets a 10% delay derating for the late arrivals of on the supply nets VDD and VDD1:

```
pt_shell> set_cross_voltage_domain_analysis_guardband -late 0.1 \
          [get_supply_nets {VDD VDD1]
```

When the **timing_enable_cross_voltage_domain_analysis** variable is set to **true**, cross-voltage-domain analysis is performed; the late delays of all cells supplied by the supply nets VDD and VDD1 are multiplied by 1.1. If ordinary derating is also specified with the **set_timing_derate** command, then both derating factors are applied to the calculated delays.

SEE ALSO

```
set_timing_derate(2)
timing_enable_cross_voltage_domain_analysis(3)
```

set_data_check

Sets data-to-data checks using the specified values of setup and hold time.

SYNTAX

```
string set_data_check
   -from from_object
      | -rise_from from_object
      | -fall_from from_object
   -to to_object
      | -rise_to to_object
      | -fall_to to_object
   [-setup | -hold]
   [-clock clock_object]
   [check_value]
```

Data Types

from_object	collection
to_object	collection
clock_object	collection
check_value	float

ARGUMENTS

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. Both rising and falling delays are checked. You must specify either the *-from*, *-rise_from*, or *-fall_from* option.

-rise_from *from_object*

Similar to the *-from* option, but applies only to rising delays at the related pin. You must specify either the *-from*, *-rise_from*, or *-fall_from* option.

-fall_from *from_object*

Similar to the *-from* option, but applies only to falling delays at the related pin. You must specify either the *-from*, *-rise_from*, or *-fall_from* option.

-to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. Both rising and falling delays are constrained. You must specify either the *-to*, *-rise_to*, or *-fall_to* option.

-rise_to *to_object*

Similar to the *-to* option, but applies only to rising delays at the constrained pin. You must specify either the *-to*,

-rise_to, or *-fall_to* option.

-fall_to *to_object*

Similar to the *-to* option, but applies only to falling delays at the constrained pin. You must specify either the *-to*, *-rise_to*, or *-fall_to* option.

-setup

Indicates that the data check value is for setup analysis only. If neither the *-setup* nor *-hold* option is specified, the value applies to both setup and hold.

-hold

Indicates that the data check value is for hold analysis only. If neither the *-setup* nor *-hold* option is specified, the value applies to both setup and hold.

-clock *clock_object*

Specifies the name of a single clock to be used for the data check. Use this option only if your design has multiple clocks per register and you want to select a single clock to be used. For more details about multiple clocks per register, see the DESCRIPTION section.

check_value

Specifies the value of the setup and/or hold time for the check.

DESCRIPTION

The **set_data_check** command specifies a data-to-data check to be performed between the from object and to object using the specified setup and/or hold value.

The pulse relation between clocks of related pin and constrained pin is considered to be zero cycles. The normal sequential check uses one cycle.

The data check is treated as non-sequential; that is, the path goes through the related and constrained pins and is not broken at these pins. To report the constraint related to the data check, use the **report_timing -to** *data_check_constrained_pin* command.

The data signals arriving at the constrained pin could come from startpoints with multiple clocks, in one of these ways:

1. Any given latch has multiple clocks propagating to the clock or gate pin.
2. When each latch has only one clock propagating to the clock or gate pin, different latches have different clocks propagating to their clock or gate pins. For such cases, these clock relations are checked separately. Similarly, data signals arriving at the related pin could also come from startpoints with multiple clocks. You use the *-clock* option to select a specific clock at the related pin for the analysis. If a single clock is not selected, multiple clocks are analyzed and grouped into respective clock groups.

Even though PrimeTime does not complain when the user defines a data-to-data check in the clock network (mainly because the tool has not yet analyzed what is clock and what is data,) the application and correct reporting of these checks may be incorrect. Data-to-data checks expect to operate on two data signals and not expected to be used in lieu of clock skew checks. So, for correct, supported behavior, the user should refrain from setting data-to-data checks in the clock network.

Note that if best-case/worst-case (bc_wc) operating conditions are preset, the same operating type is used for

both the constrained and the reference data paths. That is, when determining the setup slack for a data-to-data check, PrimeTime uses the max operating condition for both data paths. For sign-off, the user is encouraged to use on-chip variations operating mode to resolve data-to-data check setup and hold slacks.

To remove information set by the **set_data_check** command, use the **remove_data_check** command.

EXAMPLES

The following example creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, using a setup and hold time of 0.4.

```
pt_shell> set_data_check -rise_from and1/B -to and1/A 0.4
```

The following example creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, coming from the clock domain of CK1, constraining only the falling edge of signal at and1/A, using a setup time of 0.5 and no hold check.

```
pt_shell> set_data_check -rise_from and1/B -fall_to and1/A -setup \
          -clock [get_clock CK1] 0.5
```

SEE ALSO

remove_data_check(2)
report_timing(2)
update_timing(2)

set_design_attributes

Sets the specified attributes and their value on specified cells.

SYNTAX

```
string set_design_attributes
   [-elements list]
   [-attribute name_value_pair]
   [-models model_list]
```

Data Types

list	list
name_value_pair	string
model_list	list

ARGUMENTS

-elements *list*

Specifies the collection of cells on which the attributes must be set. Wildcards are accepted in cell names.

-attribute *name_value_pair*

Specifies the name and value of the attribute to be set on the cells specified by the **-elements** option. This option can be repeated multiple times to specify multiple attributes for the same set of cells in a single command.

-models *model_list*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

DESCRIPTION

This command sets the attributes and their value on a list of cells specified by the **-elements** option.

Attributes can be set multiple times on the same cells. The latest value prevails.

PrimeTime supports a limited number of attributes. For attributes that are unsupported, PrimeTime ignores them and issues a warning message.

EXAMPLES

The following example sets the **merge_domain** attribute to **true** on the mid1 cell.

```
prompt> set_design_attributes -elements {mid1} -attribute merge_domain TRUE
1
```

SEE ALSO

```
report_attribute(2)
set_port_attributes(2)
```

set_design_top

Sets or gets the current design in PrimeTime. It is a synonym for the **current_design** command.

SYNTAX

```
status set_design_top
   [design_name]
```

Data Types

```
design_name    string
```

ARGUMENTS

design_name

Specifies the working or focal design for many PrimeTime commands. If the *design_name* option is not specified, the **current_design** command returns a collection containing the current design. If the *design_name* option refers to a design that cannot be located, an error is issued and the working design remains unchanged.

DESCRIPTION

The UPF specification defines the **set_design_top** command that is in PrimeTime - implemented as a synonym to the **current_design** command.

Note: The **current_design** command removes all assertions (such as SDC, SPEF, and UPF). Therefore, the **set_design_top** command cannot be arbitrarily used in UPF scripts. It has to be used as the first command in UPF script, which is also used before any SDC or other assertions.

SEE ALSO

current_design(2)

set_disable_clock_gating_check

Disables the clock gating check for specified objects in the current design.

SYNTAX

```
string set_disable_clock_gating_check
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Specifies a list of cells, pins lib-cells or lib-pins for which the clock gating check is to be disabled.

DESCRIPTION

Disables the clock gating check for specified cells and pins in the current design. This command will only disable auto-inferred clock gating checks. Clock gating checks from library will not be disabled.

Any cell or pin in the current design or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/pin_name*.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the checking through a cell is disabled, all gating checks in the cell are disabled. When the checking through a pin is disabled, any gating check is disabled if it uses the disabled pin as a gating clock pin or a gating enable pin.

To restore gating checks disabled by the **set_disable_clock_gating_check** command, use the **remove_disable_clock_gating_check** command. To globally disable all clock gating checks, set the **timing_disable_clock_gating_checks** variable with the *true* option.

EXAMPLES

The following example disable gating checks on the specified cell, pin, lib-cell or lib-pin.

```
pt_shell> set_disable_clock_gating_check an2/Z

pt_shell> set_disable_clock_gating_check or1

pt_shell> set_disable_clock_gating_check [get_lib_cells {"class/AND1"}]

pt_shell> set_disable_clock_gating_check [get_lib_pins {class/AND1/A}]
```

SEE ALSO

```
remove_disable_clock_gating_check(2)
timing_disable_clock_gating_checks(3)
```

set_disable_timing

Disables timing arcs in a circuit.

SYNTAX

```
status set_disable_timing
   [-from from_pin_name]
   [-to to_pin_name]
   object_list
```

Data Types

from_pin_name	string
to_pin_name	string
object_list	list

ARGUMENTS

-from *from_pin_name*

Specifies that arcs only from this pin on the specified cell are disabled. The *from_pin_name* value must be a valid library pin name corresponding to the cell in the *object_list* value. When used, the **-from** and **-to** options must be specified together. The *object_list* value must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are disabled.

-to *to_pin_name*

Specifies that arcs only to this pin on the specified cell are disabled. The *to_pin_name* value must be a valid library pin name corresponding to the cell in the *object_list* value. When used, the **-from** and **-to** options must be specified together. The *object_list* value must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are disabled.

object_list

Specifies a list of cells, pins, lib-cells, lib-pins, ports, or timing-arcs that are disabled. This list can include library objects.

DESCRIPTION

Disables timing through the specified cells, pins, ports, or timing arcs in the **current_design** command.

Any cell, pin, port, or timing arc in the **current_design** command or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name*. Pins at lower levels in the

design hierarchy are specified as *instance1/instance2/cell_name/pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1/instance2/cell_name/port_name*. Timing arcs have to be collected using the *get_timing_arcs* command.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the timing through a cell is disabled, only those cell arcs are disabled that lead to the output pins of the cell. When the timing through a pin or port is disabled, only those cell arcs that lead from a load pin and to a driver pin are disabled. For bidirectional pins or ports the cell arcs from the load side and to the driver side are disabled.

When the **-from** and **-to** option pins are specified, all arcs between these two pins on the cell are disabled.

To view disabled timing arcs in the current design, use the **report_disable_timing** command. To restore timing arcs disabled by the **set_disable_timing** command, use the **remove_disable_timing** command. The **reset_design** command removes all user-specified attributes from the current design, including those set by the **set_disable_timing** command.

EXAMPLES

Consider the following reported path:

```
pt_shell> report_timing -input_pins

****************************************
Report : timing
        -path full
        -delay max
        -input_pins
        -max_paths 1
Design : counter
 ...
****************************************


  Startpoint: ffb (rising edge-triggered flip-flop)
  Endpoint: CO (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------------
  ffb/CP (FD3)                             0.00       0.00 r
  ffb/QN (FD3)                             2.42       2.42 r
  m/C (AN4)                                0.00       2.42 r
  m/Z (AN4)                                1.12       3.54 r
  CO/A (AN2)                               0.00       3.54 r
  CO/Z (AN2)                               0.48       4.02 r
  CO (out)                                 0.00       4.02 r
  data arrival time                                   4.02
  -----------------------------------------------------------------
  (Path is unconstrained)
```

Assuming you want to disable timing arcs through the CO cell, you can view all timing arcs of the CO cell by examining its library cell, AN2.

```
pt_shell> report_lib -timing lsi_10k { AN2 }

****************************************
Report : library
```

```
        -timing_arcs
Library: ...
 ...
****************************************

Library Cells:

   Attributes:
    b - black box (function unknown)
    d - dont_touch
    s - state table
    u - dont_use
    A - abstracted timing model
    E - extracted timing model
    I - Interface timing spec model (ITS)
    S - Stamp timing model
    Q - Quick timing model (QTM)

                            Arc                    Arc Pins
   Lib Cell  Attributes    #  Type/Sense      From        To         When
   ---------------------------------------------------------------------------
   AN2                     0  positive_unate  A           Z
                           1  positive_unate  B           Z
```

To disable the timing arc used in the preceding reported path, use the **set_disable_timing** command:

```
pt_shell> set_disable_timing -from A -to Z [ get_cells { CO } ]

pt_shell> report_timing

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
 ...
****************************************

  Startpoint: ffa (rising edge-triggered flip-flop)
  Endpoint: QA (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -------------------------------------------------------------
  ffa/CP (FD2)                             0.00       0.00 r
  ffa/Q (FD2)                              1.42       1.42 f
  QA/Z (IV)                                0.38       1.80 r
  QA (out)                                 0.00       1.80 r
  data arrival time                                   1.80
  -------------------------------------------------------------
  (Path is unconstrained)
```

Alternatively, the same arc is disabled as follows:

```
pt_shell> set_disable_timing [get_timing_arcs -from { CO/A } -to { CO/Z } ]
```

SEE ALSO

get_timing_arcs(2)

```
remove_disable_timing(2)
report_disable_timing(2)
report_lib(2)
report_timing(2)
reset_design(2)
timing_disable_cond_default_arcs(3)
```

set_distributed_parameters

Configures the distributed environment.

SYNTAX

```
status set_distributed_parameters
    [-collection_levels collection_level]
    [-script script]
    [-worker_common_user_data worker_script]
```

Data Types

```
collection_level    string
script              string
worker_script       string
```

ARGUMENTS

-script *script*

The user-defined/custom wrapper script to be used when launching remote processes. If this option is not specified, the wrapper is taken by default to be $synopsys_root/bin/pt_shell.

-collection_levels *collection_level*

Specifies the number of collection levels to traverse when retrieving collection attributes from the slaves. The higher this collection level is set, the more memory is potentially consumed at the master.

-worker_common_user_data *worker_script*

Specifies the filename of the Primetime script to be sourced at the startup of each worker process. The script may contain definitions of common Tcl procedures, setting of common variables and so on. The script is run after sourcing .synopsys_pt.setup files and is not intented to replace .synopsys_pt.setup.

DESCRIPTION

The **set_distributed_parameters** command allows you to configure the distributed environment. The command should be issued before the **start_hosts** command for your configurations to take effect.

EXAMPLES

In the following example one slave process is allocated on a host called platinum1. The process is launched using the **start_hosts** command.

```
pt_shell> set_host_options -num_processes 1 platinum1
1
pt_shell> set_distributed_parameters -script /usr/pt_shell
1
pt_shell> start_hosts
1] Launched: rsh -n -l user platinum1 /usr/pt_shell
             -env_start ...
             ...
               Status: Forking successful
               Stdout: Process group is (4492)
               Stderr: **<<EMPTY>>**

  ----------------------------------------------------------------------------------------
  Distributed farm creation timeout : 21600 seconds
  Waiting for   1 (of   1) distributed hosts (Mon Dec  9 05:04:36 2013)
  Waiting for   0 (of   1) distributed hosts (Mon Dec  9 05:04:46 2013)
  ----------------------------------------------------------------------------------------
```

In the following example, the **-collection_levels** option is used to restrict the amount of information returned by the **get_distributed_variable** command. By setting the collection level to *4* only four levels of collections can be traversed.

```
pt_shell> set_distributed_parameters -collection_level 4
pt_shell> remote_execute { set m_pins [get_pins A]}
pt_shell> get_distributed_variables m_pins -attr {clocks sources}
pt_shell> echo [set clock1 [get_attribute $m_pins(scen1) clocks]]
_sel12
pt_shell> echo [set clock_source1 [get_attribute $clock1 sources]]
_sel13
pt_shell> echo [set clock2 [get_attribute $clock_source1 clocks]]
_sel14
pt_shell> echo [set clock_source2 [get_attribute $clock2 sources]]
_sel15
pt_shell> echo [set clock3 [get_attribute $clock_source2 clocks]]
Warning: Attribute 'clocks' does not exist on port 'CLK' (ATTR-3)
```

In the example below, the **-worker_common_user_data** option specifies the script file to be sourced at the start of each worker process:

```
pt_shell> set_distributed_parameters -worker_common_user_data /home/user/set_worker_env.tcl
```

SEE ALSO

set_host_options(2)
start_hosts(2)

set_domain_supply_net

Sets the primary power net and primary ground net of an existing power domain.

SYNTAX

```
int set_domain_supply_net
   domain_name
   -primary_power_net supply_net_name
   -primary_ground_net supply_net_name
```

Data Types

```
domain_name          string
supply_net_name      string
```

ARGUMENTS

domain_name

Specifies the name of the power domain to set.

If there is no such power domain with the specified name in the current scope, the command fails.

-primary_power_net *supply_net_name*

Specifies the power net of the power domain.

If there is no such supply net with the specified name in the current scope, the command fails. If the supply net is not associated with the specified power domain, the command also fails.

-primary_ground_net *supply_net_name*

Specifies the ground net of the power domain.

If there is no such supply net with the specified name in the current scope, the command fails. If the supply net is not associated with the specified power domain, but is associated with other power domains, the command also fails.

DESCRIPTION

The **set_domain_supply_net** command enables you to set the primary power net and the primary ground net of a power domain. All extent of the power domains shares the same power/ground supply until explicitly excluded. Here "explicitly excluded" means it is explicitly connected with another supply net (non primary power/ground

supply net) by the following commands: **connect_supply_net**, **set_isolation**, **set_retention**. This command fails if the domain already has a primary power and ground net.

The supply net must be created in the same power domain first before it is set as the primary power/ground supply net. Use the **create_supply_net** command to create a supply net at a specified power domain.

The command returns 1 if is is successful, and returns 0 otherwise.

EXAMPLES

The following example creates two supply_net on power_domain PD1, then sets them as primary power/ground net.

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VSS -domain PD1
A_VSS
prompt> set_domain_supply_net PD1 -primary_power_net A_VDD \
-primary_ground_net A_VSS
1
```

SEE ALSO

```
create_power_domain(2)
create_supply_net(2)
connect_supply_net(2)
```

set_dont_override

Specifies the objects for which user-defined budgeted constraints should not be automatically overridden during the HyperScale analysis flow. This command is only effective during block-level analyses.

SYNTAX

```
string set_dont_override
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Specifies the design or ports for which boundary budgeted constraints should be maintained without automatic override.

DESCRIPTION

This command is used to select the ports or the entire block for which the boundary timing context pushed down from parent level analysis should not be used to override user-defined budgets.

In the HyperScale analysis flow, a top-level run pushes accurate and up-to-date boundary timing and noise context for the subblocks represented by abstraction; the subsequent block-level run automatically loads the latest timing context for the block boundary and override user-defined I/O context to perform the block level analysis. These boundary timing context for the input and output ports override user-defined I/O timing constraints, such as **set_input_delay**, **set_output_delay**, **set_input_transition**, **set_clock_latency**, and **set_input_noise**.

In a design flow, sometimes you might want the block-level design to meet a fixed timing budget prescribed for the block, instead of allowing PrimeTime to automatically override with actual timing context, which can represent some intermediate state of the parent level or other surrounding blocks. Particularly in the early design flow, or for certain portion of the design with known and clear timing specification.

You can use this command only when you also use the **set_hier_config -parent** command, meaning that the current run is for a block, and **hier_enabled_analysis** is set to **true**.

EXAMPLES

Assuming a design with two levels of hierarchy, where TOP instantiates a subblock BOT, and the named scenario of interest is fast_func. To perform hierarchical timing analysis for BOT and automatically generate a reduced and accurate timing representation for its parent level analysis, use the following commands:

```
pt_shell> set hier_enable_analysis true
pt_shell> set_hier_config -parent -name "fast_func" -path /design/TOP
pt_shell> set_hier_config -name "fast_func" -path /design/blocks/BOT
```

In addition, for the block BOT, it is necessary to freeze the timing budget defined for the bused port DATAIN so that PrimeTime does not automatically use the actual context pushed from TOP to override your settings, use the following command:

```
pt_shell> set_dont_override [get_port DATAIN*]
```

SEE ALSO

remove_dont_override(2)
link_design(2)
report_constraint(2)
save_session(2)
set_hier_config(2)
update_timing(2)
hier_enable_analysis(3)

set_dont_touch

Sets the **dont_touch** attribute on cells, nets, designs, and library cells to prevent synthesis from replacing or modifying them during optimization.

SYNTAX

```
status set_dont_touch
   object_list
   [value]
```

Data Types

```
object_list    list
value          Boolean
```

ARGUMENTS

object_list

Specifies a list of cells, nets, designs, or library cells in which to place the **dont_touch** attribute.

value

Specifies the value in which to set the **dont_touch** attribute. Allowed values are **true** (the default) or **false**.

DESCRIPTION

Sets the **dont_touch** attribute on cells and nets in the current design, or on designs and library cells, to prevent these objects from being modified or replaced by Design Compiler during optimization.

The **dont_touch** attribute is characterized by the **characterize_context** command and exported to synthesis using the **write_context** command.

When the **dont_touch** attribute on a design or library cell is written to scripts, the target objects vary based on the script format. A **dont_touch** design is applied to a design for PrimeTime and to a reference for Design Compiler. Similarly, a **dont_touch** library cell is applied to a lib_cell for PrimeTime, and to a reference for Design Compiler.

If a **dont_touch** attribute is on a design that is removed, the attribute continues to apply to any cells that were instances of that design until the next time the current design is linked.

The **dont_touch** attribute on cells, nets, and designs is imported from db files.

For a complete description of the **dont_touch** attribute and its effect, see the **set_dont_touch** command man pages in Design Compiler.

Inheritance

The **dont_touch** attribute is inherited by cells if the cell is an instance of a design or library cell that has the **dont_touch** attribute. In these cases, the **get_attribute** command returns *true* for the **dont_touch** attribute for that cell. This affects the result of the **get_attribute** command but does not actually transfer the **dont_touch** attribute to the cell. So the output of the **write_script** command does not contain extra **set_dont_touch** commands applied to cells. See the EXAMPLES section for examples.

Limitations

If your database contains a **dont_touch** attribute on an object, and you explicitly use the **set_dont_touch** command to remove the attribute by setting the value to *false* for that object, the output of the **write_script** command is inaccurate because this information is not written.

EXAMPLES

The following commands specify not to modify the "block1" and "analog1" cells during synthesis, but allow the net "N1" to be modified.

```
pt_shell> set_dont_touch [get_cells {block1 analog1}]
1
pt_shell> set_dont_touch  [get_nets N1] false
1
```

In the following example, applying a **dont_touch** attribute to design d1 affects its instances in that the **get_attribute** command returns *true* for the **dont_touch** attribute. Note the difference in the **write_script** command output for PrimeTime and Design Compiler. Note that the **write_script** command output is excerpted.

```
pt_shell> get_attribute [get_cells u1] ref_name
d1

pt_shell> get_attribute [get_cells u1] dont_touch
false

pt_shell> set_dont_touch [get_designs d1]
1

pt_shell> get_attribute [get_cells u1] dont_touch
true

pt_shell> write_script
set_dont_touch [get_designs "design1.db:d1"]

pt_shell> write_script -format dcsh
set_dont_touch find(reference, "d1")
```

SEE ALSO

```
characterize_context(2)
get_attribute(2)
```

```
set_dont_touch_network(2)
write_context(2)
write_script(2)
```

set_dont_touch_network

Sets the dont_touch_network attribute on clocks, pins, or ports in the current design to prevent cells and nets in the transitive fanout of the set_dont_touch_network objects to prevent synthesis from replacing or modifying them during optimization.

SYNTAX

```
status set_dont_touch_network
  object_list
```

Data Types

```
object_list      list
```

ARGUMENTS

object_list

Specifies a list of clocks, pins, or ports.

DESCRIPTION

Sets the **dont_touch_network** attribute on clocks, pins, or ports in the current design. It also assigns **dont_touch** attributes to all cells and nets in the transitive fanout of the **dont_touch_network** objects so that they are not modified or replaced during optimization.

The **set_dont_touch_network** command is intended primarily for clock circuitry. Placing a **dont_touch_network** attribute on a clock object prevents synthesis from modifying the clock buffer network.

This attribute is characterized by the **characterize_context** command and exported to synthesis using the **write_context** command.

If you specify a pin or a port, all nets and cells in its transitive fanout are recursively marked **dont_touch**. This recursion propagates through combinational cells but stops at registers (without marking the registers as **dont_touch**). An element is recognized as a register only if it has setup or hold constraints. If you specify an input pin, its own cell is marked **dont_touch** but its connected net is not marked **dont_touch**. If you specify an output pin, its own cell is not marked **dont_touch** but its connected net is marked **dont_touch**.

Refer to the **set_dont_touch** command man page in Design Compiler for a complete description of the **dont_touch** attribute and its effect.

Use the **reset_design** command to remove the **dont_touch_network** and **dont_touch** attribute.

EXAMPLES

The following command adds a **dont_touch_network** attribute to the port named "clock_in".

```
pt_shell> set_dont_touch_network [get_ports clock_in]
```

The following command adds the **dont_touch** attribute to a net named net1 and a cell named cell2 but not to the cell named cell1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> set_dont_touch_network [get_pins cell1/Y]
1
```

The following command adds the **dont_touch** attribute to a net named net1 and cell named cell1 but not on the net named net0 and register reg1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y reg1/D}
prompt> get_pins -of_objects [get_cells cell1]
{cell1/A cell1/Y}
prompt> get_nets -of_objects [get_pins cell1/A]
{net0}
prompt> set_dont_touch_network [get_pins cell1/A]
1
```

SEE ALSO

```
characterize_context(2)
report_context(2)
set_dont_touch(2)
write_context(2)
```

set_dont_use

Sets the **dont_use** attribute on library cells to exclude them from the target library during optimization.

SYNTAX

```
status set_dont_use
   object_list
   [true | false]
```

Data Types

```
object_list    list
```

ARGUMENTS

object_list

Specifies a list of objects (library cells, modules, or implementations) on which the **dont_use** attribute is to be set.

true | false

Specifies whether the **dont_use** attribute is set to **true** (the default) or **false**.

DESCRIPTION

This command sets the **dont_use** attribute on specified library objects, so that they are not used for ECO. The specified objects must be in one of the target libraries or in a library already loaded into memory.

use this command with care, especially when changing the value of the **dont_use** attribute from **true** to **false**. Follow the design requirements that establish which library cells can be used for ECO. Disabling library cells can have an impact on PrimeTime ECO results depending on the choices of library cells available for ECO.

This command affects only the copy of the library that is currently loaded into memory and has no effect on the version that exists on disk.

In the distributed multi-scenario analysis flow, you should apply the **set_dont_use** command in all the scenarios so that the specified library cells are not used during optimization, such as when using the **fix_eco_timing** command.

EXAMPLES

Setting the **dont_use** attribute to **true** on a library cell disables it from ECO usage while setting it to **false** enables it for ECO usage.

The following command sets the **dont_use** attribute on the G1 and G2 lib_cells in the mylib library:

```
pt_shell> set_dont_use {mylib/G1 mylib/G2}
1
```

The following command unsets the **dont_use** attribute on the G1 lib_cell in the mylib library:

```
pt_shell> set_dont_use {mylib/G1} false
1
```

SEE ALSO

```
fix_eco_drc(2)
fix_eco_power(2)
fix_eco_timing(2)
get_attribute(2)
report_lib(2)
```

set_drive

Sets the resistance to a specified value on specified input or inout ports in the current design.

SYNTAX

```
status set_drive
   [-rise]
   [-fall]
   [-min]
   [-max]
   resistance_value
   port_list
```

Data Types

resistance_value	float
port_list	list

ARGUMENTS

-rise

Specifies that the *resistance_value* argument is used to drive the ports only for the rising case.

-fall

Specifies that the *resistance_value* argument is used to drive the ports only for the falling case.

-min

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* argument is the minimum resistance.

-max

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* argument is the maximum resistance.

resistance_value

Specifies a nonnegative port drive resistance value for the ports in the *port_list*. The value must be >= 0; units must be the same as those in the logic library.

port_list

Specifies a list of input or inout ports in the current design, in which the *resistance_value* argument is set.

DESCRIPTION

This command sets the resistance to a specified value on specified input or inout ports in the current design. PrimeTime models the driver as a resistance and uses that value to calculate the wire delay of the port. To view the drive that is set, use the **report_port -drive** attribute.

Depending on the options used, the **set_drive** command sets these attributes on the specified ports:

```
drive_resistance_fall_max
drive_resistance_fall_min
drive_resistance_rise_max
drive_resistance_rise_min
```

If the **set_drive** command is issued without any options, the **drive_resistance_fall_max** and **drive_resistance_rise_max** attributes are set; in min-max mode, the other two attributes are also set.

If the **set_drive** command is issued with only the **-rise** option, only the **drive_resistance_rise_max** attribute is set; in min-max mode, the **drive_resistance_rise_min** attribute is also set.

If the **set_drive** command is issued with only the **-fall** option, only the **drive_resistance_fall_max** attribute is set; in min-max mode, the **drive_resistance_fall_min** attribute is also set.

Drive resistance is the output resistance of the cell that drives the port, so that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance_value* of 0 is infinite drive, or no delay between the ports and all ports connected to them. Setting *resistance_value* to 0 is the same as removing the drive resistance with the **remove_drive_resistance** command.

There are two other methods of describing port drive capability. **set_driving_cell** command or the **set_input_transition** command. The most recent drive command has precedence.

EXAMPLES

The following example sets the drive resistance to 5 on all input ports in the current design, and displays the ports on which the drive resistance has been set.

```
pt_shell> set_drive 5 [all_inputs]
1
pt_shell> report_port -drive
****************************************
Report : port
        -drive
Design : counter
****************************************

                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
-------------------------------------------------------------
A             5.00      5.00      --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --
CL            5.00      5.00      --        --
CLK           5.00      5.00      --        --
D             5.00      5.00      --        --
JOKE          5.00      5.00      --        --
L             5.00      5.00      --        --
P             5.00      5.00      --        --
```

```
RESET          5.00      5.00      --        --
T              5.00      5.00      --        --
```

SEE ALSO

remove_drive_resistance(2)
report_port(2)
set_load(2)

set_drive_resistance

Sets drive resistance for input or inout ports.

Note: This command is obsolete, and has been replaced by the **set_drive** command. Use the **set_drive** command instead.

SYNTAX

```
string set_drive_resistance
   [-rise]
   [-fall]
   [-min]
   [-max]
   resistance
   port_list
```

Data Types

resistance	float
port_list	list

ARGUMENTS

-rise

Specifies to use the resistance to drive the ports for the rising case.

-fall

Specifies to use the resistance to drive the ports for the falling case.

-min

Specifies to use the resistance to drive the ports for the minimum case.

-max

Specifies to use the resistance to drive the ports for the maximum case.

resistance

Specifies a nonnegative resistance value for the ports. Port drive resistance (value >= 0).

port_list

Specifies a list of input or inout port names of the current design on which the drive attributes are to be set.

DESCRIPTION

Model the driver as a resistance and use that value to calculate the wire delay of the port. The **report_port -drive** command can be used to view the drive that is set.

Drive resistance is the output resistance of the cell that drives the port; such that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance* of 0 is infinite drive, or no delay between the ports and all connected to them. This is the same as removing the drive resistance with the **remove_drive_resistance** command.

Drive resistance must be in units with the technology library.

EXAMPLES

Here is an example of the **set_drive_resistance** command in use.

```
pt_shell> set_drive_resistance 5 [all_inputs]
1
pt_shell> report_port -drive
****************************************
Report : port
        -drive
Design : counter
****************************************

                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
-------------------------------------------------------------
A             5.00      5.00      --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --
CL            5.00      5.00      --        --
CLK           5.00      5.00      --        --
D             5.00      5.00      --        --
JOKE          5.00      5.00      --        --
L             5.00      5.00      --        --
P             5.00      5.00      --        --
RESET         5.00      5.00      --        --
T             5.00      5.00      --        --
```

SEE ALSO

remove_drive_resistance(2)
report_port(2)
set_drive(2)
set_load(2)

set_driving_cell

Sets the port driving cell.

SYNTAX

```
string set_driving_cell
   [-lib_cell lib_cell_name]
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-library lib_name]
   [-pin pin_name]
   [-from_pin from_pin_name]
   [-no_design_rule]
   [-input_transition_rise rtran]
   [-input_transition_fall ftran]
   [-clock clock_name]
   [-clock_fall]
   port_list
```

Data Types

lib_cell_name	string
lib_name	string
pin_name	string
from_pin_name	string
rtran	float
ftran	float
clock_name	string
port_list	list

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell used to drive the ports. You must use the **-pin** option if the cell has more than one output pin. If different cells are needed for the rising and the falling cases, use separate commands with the **-rise** or **-fall** option.

Use the **-from_pin** option to choose between multiple input pins with arcs to this output pin. This option is required.

-rise

Sets driving cell information for only the rising port transition.

-fall

Sets driving cell information for only the falling port transition.

-min

Sets driving cell information for only the minimum analysis. This option is valid only in min-max mode.

-max

Sets driving cell information for only the maximum analysis. This option is valid even when not in min-max mode. When not in min-max mode, the option is not required because it is the default.

-library *lib_name*

Specifies the name of the library containing the *library_cell_name* value. This is the library of the driving cell. By default, the libraries specified with the **link_path** variable are searched for the cell.

-pin *pin_name*

Specifies the output pin whose drive is used. This is the driving pin name. If you do not include the **-from_pin** option, the command uses an arbitrary pin arc ending at the specified pin.

-from_pin *from_pin_name*

Specifies an input pin on the specified cell so the command uses the drive of the timing arc from this pin to the specified pin.

-no_design_rule

Prevents the transfer of design rules from the driving cell to the port. If you do not include this option, the design rules (such as max_capacitance) of the library pin are applied to the port.

-input_transition_rise *rtran*

Specifies the input rising transition time associated with the **-from_pin** option; if you do not include this option, the default is 0. Use the **-input_transition_rise** and **-input_transition_fall** options to capture the accurate transition time associated with the *from_pin_name* value. This can obtain more accurate information on the transition time and delay time at the output pin.

-input_transition_fall *ftran*

Specifies the input falling transition time associated with the *from_pin_name* value; if you do not include this option, the default is 0.

-clock *clock_name*

Specifies the clock related to the driving cell. This option is deprecated and ignored by the tool.

-clock_fall

Specifies that driving cell is relative to the falling edge of the clock. The default is the rising edge. This option is deprecated and ignored by the tool.

port_list

Provides a list of input ports. The list contains input or inout port names in the current design on which the driving cell information is set.

DESCRIPTION

The **set_driving_cell** command sets the port driving cell. It sets attributes on the specified input or inout ports in the current design to associate an external driving cell with the ports. The drive capability of the port is the same as if the specified driving cell were connected in the same context to allow accurate modeling of port drive capability for nonlinear delay models. Use the **report_port** command with the **-drive** option to view drive information on ports.

To automatically set driving cell information on subdesign ports based on their context in the entire design, use the **characterize_context** command. To remove driving cell information from ports, use the **remove_driving_cell** command.

Note: There are two other methods of describing port drive capability, using the **set_drive** or **set_input_transition** command. The most recent drive command has precedence. If possible, always use the **set_driving_cell** command instead of the **set_drive** command because the **set_driving_cell** command allows accurate calculation of port delay and transition time for library cells with nonlinear dependence on capacitance.

EXAMPLES

The following example shows how to use the **set_driving_cell** command, and shows how to view drive information on ports using the **report_port** command.

```
pt_shell> set_driving_cell -lib_cell AND [get_ports A]

pt_shell> report_port -drive A

****************************************
Report : port
        -drive
Design : counter
 ...
****************************************

                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
------------------------------------------------------------
A             --        --        --        --


                     Driving Cell
Input Port    Rise                Fall         Mult    Attrs
-------------------------------------------------------------
A             AND                 AND          --
```

SEE ALSO

```
all_inputs(2)
characterize_context(2)
remove_driving_cell(2)
report_port(2)
reset_design(2)
set_drive(2)
set_load(2)
set_input_transition(2)
```

set_effective_voltage_options

Set options to be used in calculating, reporting and map displaying effective voltages.

SYNTAX

```
status set_effective_voltage_options
   [-type type]
   [-mode mode]
```

Data Types

```
string type
string mode
```

ARGUMENTS

-type *type*

Specifies which type of effective voltages to be calculated during rail analysis. Available options are *default* and *switching_window*. The default type is *default*, which means effective voltage is calculated cross the whole simulation window during rail analysis. When *switching_window* is specified, effective voltage is only calculated within the windows where the cell is switching.

-mode *mode*

Specifies which kind of effective voltage value is reported in the command **write_rail_data** and shown in the effective voltage map display. Available options are MIN, MAX and AVG, where MIN means minimum effective voltage, MAX means maximum effective voltage, and AVG means average effective voltage. The default value is MAX if this option is never specified.

DESCRIPTION

The **set_effective_voltage_options** command is to set options for calculating, reporting, and map displaying effective voltage. The command returns 1 if successful, 0 if unsuccessful.

This command has to used before the command **calculate_rail_voltage**. This is where effective voltage is calculated. After rail analysis is done, this command can be used to choose which kind of effective voltage values to be reported in the command **write_rail_data** with the option *-type effective_voltage_drop_or_rise*, and during GUI map display.

A typical usage of the command is to first specify how effective voltage should be calculated during **calculate_rail_voltage** using the option *-type*. User can choose to calculate the effective voltage cross the

whole simulation window, which is the default behavior even without using the command. Or if user prefers, he can choose the options *-type default*. If user wants to calculate the effective voltage only within windows where the cell is switching, he can specify the option *-type switching_window*. Note, not all cells will be switching during rail analysis. In this case, the effective voltage will be calculated as zero for cells which are not switching at all during whole simulation window. During rail analysis, all three different kinds of effective voltage values will be calculated. There are minimum (MIN), maximum (MAX), and average (AVG) values available after rail analysis is done.

Once the user sets how effective voltage should be calculated during **calculate_rail_voltage**, he then can choose what kind of values to write out and also show in GUI map display. User can invoke the command with the option *-mode* to select which kind of effective voltage values to be reported in the command **write_rail_data** with option *-effective_voltage_drop_or_rise*. The subsequent GUI map display will also be reflected with this selction.

EXAMPLES

In the following example, user does not use the command **set_effective_voltage_options** before invoking **calculate_rail_voltage**. By default the effective voltage is calculated cross the whole simulation window. Then the user uses the command to set which value to be reported and generates three different kinds of reports.

```
pr_shell> calculate_rail_voltage {VDD VSS} -dynamic
pr_shell> write_rail_data -type effective_voltage_drop_or_rise max.rpt
pr_shell> set_effective_voltage_options -mode MIN
pr_shell> write_rail_data -type effective_voltage_drop_or_rise min.rpt
pr_shell> set_effective_voltage_options -mode AVG
pr_shell> write_rail_data -type effective_voltage_drop_or_rise avg.rpt
```

The following example sets effective voltage to be calculated from switching windows and then report.

```
pr_shell> set_effective_voltage_options -type switching_window
pr_shell> calculate_rail_voltage {VDD VSS} -dynamic
pr_shell> set_effective_voltage_options -mode MAX
pr_shell> write_rail_data -type effective_voltage_drop_or_rise window_max.rpt
pr_shell> set_effective_voltage_options -mode MIN
pr_shell> write_rail_data -type effective_voltage_drop_or_rise window_min.rpt
pr_shell> set_effective_voltage_options -mode AVG
pr_shell> write_rail_data -type effective_voltage_drop_or_rise window_avg.rpt
```

SEE ALSO

calculate_rail_voltage(2)

set_em_options

Specifies configuration information for electromigration analysis.

SYNTAX

```
string set_em_options
   [-temperature temp_value]
   [-snap_temperature_to_rule snap_value]
   [-scale_factor scale_value]
   [-width_limit width_value]
   [-relative_width_limit percentage_value]
```

Data Types

```
float temp_value
int   snap_value
float scale_value
float width_value
float percentage_value
```

ARGUMENTS

-temperature *temp_value*

Specifies the temperature value, in degrees Celsius, to be used for evaluating electromigration constraints. The allowed values are -300.0 through +500.0. By default, the command uses the nominal temperature of the default operating condition in the first library in the link path.

-snap_temperature_to_rule *snap_value*

Setting the *snap_value* to 1 specifies the nearest temperature value in the electromigration rules to be used when the user-specified temperature_value does not match the defined temperatures in the electromigration rule tables. By default, the *snap_value* is 0 and interpolation is performed to derive approximate rules for the specified *temp_value*.

-scale_factor *scale_value*

Specifies a uniform scale factor to be applied to the current density values in the electromigration rules before testing for electromigration violations. The default value is 1.0.

-width_limit *width_value*

Specifies an absolute minimum width, in microns, for polygons to be included in the electromigration violation checking. If the width of a polygon carrying current that violates an electromigration rule is below *width_value*, it is not included in the electromigration violation report.

By default, there is no absolute minimum width for polygons to be included in the electromigration violation.

-relative_width_limit *relative_width_value*

Specifies a minimum width, as a percentage of the layer minimum width defined in the technology file, for polygons to be included in the electromigration violation checking. If the width of a polygon carrying current that violates an electromigration rule is below the defined percent of the minimum width for its layer, it is not included in the electromigration violation report.

By default, the percentage_value is set to 80 if the relative_width_limit option is not specified.

DESCRIPTION

The command specifies configuration options for electromigration analysis. The **set_em_options** command can be executed multiple times to set the options incrementally. The current electromigration options can be reported by the **report_em_options** command.

EXAMPLES

The following example specifies configuration options for electromigration violation checking:

```
pr_shell> set_em_options -temperature 125 -snap_temperature_to_rule 1 -scale 0.8
1
pr_shell> report_em_options
****************************************
Report : Options for EM checking
Design : design
Version: G-2012.06-Alpha
Date   : Tue Apr 24 15:03:57 2012
****************************************

Temperature:   125.00 C

Snap Temperature to Rule: ON

Scale Factor:    0.800

Width Limit: NONE

Relative Width Limit:    80.00 %

1
```

SEE ALSO

```
report_em_options(2)
report_em_rules(2)
report_em_violations(2)
update_em(2)
```

set_equal

Sets two ports to be logically equivalent.

SYNTAX

```
status set_equal
  port1
  port2
```

Data Types

```
port1        string
port2        string
```

ARGUMENTS

port1

Specifies the first input port.

port2

Specifies the second input port.

DESCRIPTION

Defines two input ports in the current design as logically equivalent. This information is used during synthesis to eliminate redundant ports, improving optimization quality.

Logical inconsistencies are checked for in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

To remove this property from a port, run the **reset_design** command.

This attribute is characterized by the **characterize_context** command and exported to synthesis using the **write_context** command.

EXAMPLES

The following example sets two input ports "A" and "B" logically equal.

```
pt_shell> set_equal A [get_ports B]
1
```

The following example sets up a contradictory relationship between two ports and creates an error.

```
pt_shell> set_equal A B
1
pt_shell> set_opposite A B
Error: Can't set equal ports opposite in design 'top': 'A' 'B'. (DDB-22)
0
```

SEE ALSO

```
characterize_context(2)
set_opposite(2)
write_context(2)
```

set_extract_supply_parasitics_options

Specifies configuration information needed for the **extract_supply_parasitics** command.

SYNTAX

```
set_extract_supply_parasitics_options
   [-max_tluplus filename]
   [-tech2itf_map file_name]
   [-temperature temp_value]
   [-skip_metal_layer skip_metal_file]
   [-include_shapes list_additional_shapes]
   [-via_array_partition_size {{via1 n1} {via2 n2} ...}]
   [-derive_macro_pin_shapes true | false]
   [-implicit_port_connection mw_cell_type_list]
   [-exclude_of_implicit_port_connection cell_master_name_list]
   [-place_current_node_by_distance distance_in_db_unit]
```

Data Types

```
string file_name
float range temp_value
string skip_metal_file
shape type collection list_additional_shapes
```

ARGUMENTS

-max_tluplus *file_name*

Specifies the TLUPlus technology file to be used for extraction. If this option is not specified, the tool reads the file from the location specified in the TLUPlus option that is saved in Milkyway, if present.

-tech2itf_map *file_name*

Specifies the TLUPlus layer mapping file. If this option is not specified, the tool reads the file from the location specified in the TLUPlus option that is saved in Milkyway, if present.

-temperature *temp_value*

Specifies the temperature, in degrees Celsius, to be used during supply net extraction. By default, the nominal temperature from the default operating condition in the first library in the link path is used. If this setting is given, the nominal values in the TLUPlus file are derated according to their temperature coefficients and this temperature setting. The allowed values are -300.0 through +500.0.

-skip_metal_layer *skip_metal_file*

Specifies the metal layer to be skipped. The metal layers lower than the specified layer will be skipped. You can specify the hard macros to be skipped in the *skip_metal_file*.

-include_shapes *list_additional_shapes*

Specifies the special shapes in the standard cells that need to be included during extraction. For example, setting *list_additional_shapes* to {stdcell_vias} means the vias inside standard cells need to be included for extraction.

Note: When running PrimeRail version 2013.12 and before, you must specify this option when the design has dual rail routings on the PG net. This is to ensure the internal vias on the via12 layer are extracted when both M1 pins and M2 pins of a standard cell are connected to the top-level PG rail.

Beginning with PrimeRail version 2014.06, you do not need to specify this option to include the vias inside standard cells during extraction. PrimeRail automatically extracts these vias when the design has dual rail routings.

-via_array_partition_size *{{via1 n1} {via2 n2} ...}*

Breaks a large square or rectangular via array into small via arrays based on the specified via mask name and via array dimension. The via mask name can be *ALL*, meaning all via layers are applied with the same partition size setting. By default, the via array partition size is {ALL 8}, which partitions via array by 8x8 via cuts for all via mask layers.

-derive_macro_pin_shapes true | false

Controls whether PrimeRail gets physical block instance supply pin shapes from its FRAM view (by setting to *false*) or derives all the overlaps from the block and top shapes automatically by ignoring the FRAM view pin shapes (by setting to *true*). Use this option to determine the supply pin shapes based on the supply shape overlaps between the enclosing net shapes and the physical block instance net shapes.

The default is false.

-implicit_port_connection *mw_cell_type_list*

During extraction, connects the pin shapes that are implicitly but not explicitly connected for black box cells based on the EEQ class setting.

When a cell does not have a macro model, PrimeRail uses its FRAM view as its macro model. FRAM view is an abstract model and might not model its pin shapes connection explicitly.

The *mw_cell_type_list* is a list of Milkyway cell types defined by the *mask_layout_type* attribute, including *antenna*, *chip*, *tap*, *cover*, *macro*, *soft_macro_io_fixed*, *io_pad*, *corner_pad*, *pad_filler*, *flip_chip_pad*, *flip_chip_driver*, *std*, *std_filler*, *double_height_std*, *triple_height_std*, *triple_more_height_std*, and *through_silicon_via*.

-exclude_of_implicit_port_connection *cell_master_name_list*

Excludes the specified cell masters from the implicit port connection.

-place_current_node_by_distance *distance_in_db_unit*

By default, supply net extraction only creates one current node on one pin shape. With specified this option, supply net extraction creates multiple current nodes by specified distance.

DESCRIPTION

The **set_extract_supply_parasitics_options** command specifies the setup options used by the supply net extraction command **extract_supply_parasitics**. The **set_extract_supply_parasitics_options** command can be executed multiple times to set the options incrementally. To check the current settings, run the **report_extract_supply_parasitics_options** command.

EXAMPLES

The following example loads a Milkyway design and sets the extraction options:

```
pr_shell> open_mw_cel -lib my_lib my_cel
pr_shell> set_extract_supply_parasitics_options \
   -max_tluplus /design/max.tlu \
   -tech2itf_map /design/map_file \
   -temperature  125.00 \
   -via_array_partition_size {{via1 10} {via2 10}} \
   -skip_metal_layer /design/skip_metal_file
```

SEE ALSO

extract_supply_parasitics(2)
open_mw_cel(2)
report_extract_supply_parasitics_options(2)
report_supply_parasitics(2)

set_false_path

Identifies paths in a design that are to be marked as false, so that they are not considered during timing analysis.

SYNTAX

```
status set_false_path
   [-setup]
   [-hold]
   [-rise]
   [-fall]
   [-reset_path]
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-comment comment_string]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list
comment_string	string

ARGUMENTS

-setup

Indicates that setup (maximum) paths are to be marked as false. This option disables setup checking for specified paths. If you do not specify either the **-setup** or **-hold** option, both setup and hold timing are marked false.

-hold

Indicates that hold (minimum) paths are to be marked as false. This option disables hold checking for specified paths. If you do not specify either the **-setup** or **-hold** option, both setup and hold timing are marked false.

-rise

Indicates that rising delays are to be marked as false, as measured on the path endpoint. If you do not specify either the **-rise** or **-fall** option, both rise and fall timing are marked false.

-fall

Indicates that falling delays are to be marked as false, as measured on the path endpoint. If you do not specify either the **-rise** or **-fall** option, both rise and fall timing are marked false.

-reset_path

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with only the **-to** option, all paths leading to the specified endpoints are reset. If used with only the **-from** option, all paths leading from the specified startpoints are reset. If used with the **-from** and **-to** options, only paths between those points are reset. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_false_path** command is issued.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-through *through_list*

Specifies a list of pins, ports, cells or nets through which the disabled paths must pass. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you do not specify any type of *through* option, all timing paths specified using the **-from** and **-to** options are affected.

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

-rise_through ***rise_through_list***

The same as the **-through** option, but the paths must have a rising transition at the through points.

-fall_through ***fall_through_list***

The same as the **-through** option, but the paths must have a falling transition at the through points.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-comment ***comment_string***

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

The **set_false_path** command marks startpoint/endpoint pairs as false timing paths; that is, paths that cannot propagate a signal. The command removes timing constraints on these false paths, so that they are not considered during timing analysis. Path startpoints are input ports or register clock pins; path endpoints are register data pins or output ports. The command disables maximum delay (setup) checking and minimum delay (hold) checking for the specified paths.

The **set_false_path** command is a point-to-point timing exception command, and overrides the default single-cycle timing relationship for one or more timing paths. False path information always takes precedence over multicycle path information. Also, the **set_false_path** command overrides the **set_max_delay** and **set_min_delay** commands.

In general, for timing exceptions with different path specifications, the more specific command has priority over the more general one. False paths represent an exception to this rule. If a timing path is satisfied by two or more false path specifications, the more general one wins. This can be seen with the **report_exceptions -ignored** command.

To disable the timing at a particular cell along a path, use the **set_disable_timing** command.

To remove the false path designations set by the **set_false_path** command, use the **reset_path** command.

For crosstalk analysis the false paths are treated as sensitizable. Therefore, the aggressors on the false paths could still effect its victims. However, the presence of false path could change the borrow behavior in level-sensitive latch designs, potentially borrow latches may no longer borrow leading to change in delta delays for logic nets that are in the fanout of affected latches. When the clocks are asynchronous to each other, the **set_clock_groups -asynchronous** command should be used instead of the **set_false_path** command between the asynchronous clocks. Without the **set_clock_groups -asynchronous** command, the clocks are treated to be synchronous to each other.

EXAMPLES

The following example disables timing paths from ff12 to ff34.

```
pt_shell> set_false_path -from ff12 -to ff34
```

The following example disables rising timing paths from U14/Z to ff29/Reset.

```
pt_shell> set_false_path -rise_from U14/Z -to ff29/Reset
```

The following examples disables hold checking for endpoints clocked by CLK1.

```
pt_shell> set_false_path -hold -to [get_clocks CLK1]
```

The following example disables all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}.

```
pt_shell> set_false_path -from ff1/CP -through {U1/Z U2/Z}
   -through {U3/Z U4/C} -to ff2/D
```

The following example disables all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall to one or more of {U3/Z U4/C}.

```
pt_shell> set_false_path -from ff1/CP
   -rise_through {U1/Z U2/Z}
   -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

current_design(2)
report_exceptions(2)
reset_design(2)
reset_path(2)
set_clock_groups(2)
set_disable_timing(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

set_fanout_load

Sets fanout_load for output ports in the current design.

SYNTAX

```
string set_fanout_load
   value
   port_list
```

Data Types

value	float
port_list	list

ARGUMENTS

value

Shows the **fanout_load** value, in units consistent with the **fanout_load** and **max_fanout** values in the technology library.

port_list

Provides a list of output ports.

DESCRIPTION

Specifies a **fanout_load** for output ports in the current design. By default, ports are considered to have fanout_load of 0.0. Fanout load for a net is the sum of **fanout_load** attributes for all input pins and output ports connected to the net. Output pins may have maximum fanout limits, specified in the library or with the **set_max_fanout** command.

The **report_constraint -max_fanout** command shows maximum fanout constraint evaluations. The **report_port -design_rule** command shows port **fanout_load** values.

To remove **fanout_load** information from ports, use the **remove_fanout_load** command.

EXAMPLES

This example sets the fanout_load on ports matching "OUT*" to 3.0.

```
pt_shell> set_fanout_load 3.0 "OUT*"
```

SEE ALSO

remove_fanout_load(2)
report_constraint(2)
report_port(2)
set_max_fanout(2)
set_min_fanout(2)

set_host_options

Specifies host options for compute resources.

SYNTAX - Current (master) process

```
int set_host_options
   [-local_process]
   [-max_cores number]
```

SYNTAX - Remote (slave) processes

```
int set_host_options
   [-max_cores number]
   [-name name]
   [-submit_command command]
   [-terminate_command command]
   [-num_processes number]
   [-32bit]
   [host_names]
```

Data Types

number	integer
name	string
command	string
host_names	list

ARGUMENTS

-local_process

This option is not supported in PrimeRail. It's ignored.

-max_cores *number*

Use this option to modify the CPU cores usage limit for the current run. The *number* value is currently limited to the range between 1 and the number of cores in the current machine.

-name *name*

This option is not supported in PrimeRail. It's ignored.

-submit_command *command*

This option is not supported in PrimeRail. It's ignored.

-terminate_command *command*

This option is not supported in PrimeRail. It's ignored.

-num_processes *number*

This option is not supported in PrimeRail. It's ignored.

-32bit

This option is not supported in PrimeRail. It's ignored.

host_names

This option is not supported in PrimeRail. It's ignored.

DESCRIPTION

The set_host_options command is used to specify the maximum number of CPU cores that the tool can use during parallel execution. The default is 4. The maximum number of cores that can be specified is the number of cores in the current machine.

To disable parallel execution, set the -max_cores option to 1.

SEE ALSO

```
set_distributed_parameters(2)
remove_host_options(2)
report_host_usage(2)
```

set_ideal_latency

Specifies ideal latency values for the pins in an ideal network.

SYNTAX

```
int set_ideal_latency
   [-rise] [-fall]
   [-min] [-max]
   value
   object_list
```

Data Types

value	float
object_list	list

ARGUMENTS

-rise

Indicates that the *value* option represents the rise latency time. If you do not specify the *-rise* or *-fall* options, both values are set.

-fall

Indicates that the *value* option represents the fall latency time. If you do not specify the *-rise* or *-fall* option, both values are set.

-min

Indicates that the *value* option represents the minimum latency time. If you do not specify the *-min* or *-max* option, both values are set.

-max

Indicates that the *value* option represents the maximum latency time. If you do not specify the *-min* or *-max* option, both values are set.

value

Specifies an ideal latency value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal latency is set.

DESCRIPTION

Sets an ideal latency value on leaf cell pins and top-level ports of an ideal network.

PrimeTime uses ideal timing for ideal networks, which means that pins and ports have a specified ideal latency (from the **set_ideal_latency** command) or zero ideal latency by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal latency value provides an estimate of the latency time in the ideal network for pre-layout.

The specified latency value overrides the internally-estimated cell and net delay value and any other delay annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the **report_ideal_network** command and the ideal latency *value* option is ignored for those pins. Ideal latency values do not have any effect in ideal clock networks (for example, non-propagated clock networks).

The **set_ideal_latency** command affects all pins in the transitive fanout of the pins or ports on which ideal latency is specified. The total ideal latency at an ideal boundary pin is the sum of latency on the ideal network source and all ideal latencies on the path.

You can use the **set_ideal_latency** command for pins at lower levels of the design hierarchy.

To list ideal latency values, use the **report_ideal_network** command.

To remove the ideal latency values from a design, use the **remove_ideal_latency** or **reset_design** command.

EXAMPLES

The following example specifies a rise latency of *1.2* and a fall latency of *0.9* for ports named *A*, *B*, and *C*.

```
pt_shell> set_ideal_latency 1.2 -rise {A B C}
pt_shell> set_ideal_latency 0.9 -fall {A B C}
```

SEE ALSO

```
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
report_timing(2)
set_ideal_transition(2)
set_ideal_network(2)
```

set_ideal_network

Marks a set of ports or pins in the design as sources of an ideal network. This disables timing update of cells and nets in the transitive fanout of the specified objects.

SYNTAX

```
integer set_ideal_network
   [-no_propagate]
   object_list
```

Data Types

object_list	list

ARGUMENTS

-no_propagate

Indicates that the ideal network is not propagated through leaf cells. Ideal properties are enabled on all nets that are electrically connected to the ideal network sources.

object_list

Specifies a list of objects to mark as the sources of an ideal network. Ideal network source objects may be ports or pins of leaf cells at any hierarchical level of the design. If nets are specified in the *object_list*, all of the nets' global driver pins are marked as ideal network sources. The **set_ideal_network** command accepts nets only when you specify the *-no_propagate* option.

DESCRIPTION

This command marks a set of ports or pins in the design as sources of an ideal network. You specify only the source of the network. PrimeTime marks all nets, cells, and pins in the transitive fanout of ideal sources as ideal. The ideal property is automatically spread by the tool and spread again, as necessary, after netlist changes. The criteria for propagating the ideal property, starting at the source pins and ports, are as follows:

- A pin is marked as ideal if you specify it by using the **set_ideal_network** command, if it is either a driver pin and its cell is ideal or if it is a load pin attached to a net that is ideal.
- A net is marked as ideal if all its driving pins are ideal.
- A combinational cell is marked as ideal if at least one of its input pins is ideal and all the other input pins are either ideal, unconnected or attached to a logic constant nets. Objects with the case analysis attribute set are not treated as constant.

- Propagation traverses through combinational cells but stops at sequential cells.
- Design rule checks (DRCs) are disabled on ideal network objects.
- Ideal networks can be specified in the clock or data network.

The latency and transition times of an ideal network are zero by default, but you can override them by using the **set_ideal_latency** and **set_ideal_transition** commands. Ideal timing values do not have any effect in ideal clock networks (for example, non-propagated clock networks).

To reverse the effect of the **set_ideal_network** command, use the **remove_ideal_network** command and specify the source pins or ports for the network. All ideal networks are removed using the **reset_design** command.

Ideal sources, non-source ideal pins with ideal timing values, boundary pins of ideal networks and ideal nets and cells are displayed using the **report_ideal_network** command.

EXAMPLES

The following example creates an ideal network on ports in a design called 'TOP'.

```
pt_shell> current_design {TOP}
pt_shell> set_ideal_network {IN1 IN2}
```

The following example creates an ideal network on the multi-driven net named 'netA'.

```
pt_shell> set_ideal_network -no_propagate {netA}
Warning: Transferring ideal net attribute onto driver pin 'U1/Z' of net 'netA'. (UITE-450)
Warning: Transferring ideal net attribute onto driver pin 'FFD/Q' of net 'netA'. (UITE-450)
```

SEE ALSO

```
remove_ideal_network(2)
report_ideal_network(2)
reset_design(2)
set_ideal_latency(2)
set_ideal_transition(2)
```

set_ideal_transition

Specifies ideal transition values for the pins in an ideal network.

SYNTAX

```
int set_ideal_transition
   [-rise] [-fall]
   [-min] [-max]
   value
   object_list
```

Data Types

value	float
object_list	list

ARGUMENTS

-rise

Indicates that the *value* option represents the rise transition time. If you do not specify the *-rise* or *-fall* option, both values are set.

-fall

Indicates that the *value* option represents the fall transition time. If you do not specify the *-rise* or *-fall*option, both values are set.

-min

Indicates that the *value* option represents the minimum transition time. If you do not specify the *-min* or *-max*, both values are set.

-max

Indicates that the *value* option represents the maximum transition time. If you do not specify the *-min* or *-max* option, both values are set.

value

Specifies an ideal transition value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal transition is set.

DESCRIPTION

Sets an ideal transition value on leaf cell pins and top-level ports of an ideal network.

PrimeTime uses ideal timing for ideal networks, which means that pins and ports have a specified ideal transition (from the **set_ideal_transition** command) or zero ideal transition by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal transition value provides an estimate of the transition time in the ideal network for pre-layout.

The specified transition value overrides the internally-estimated cell and net transition value and any other transition annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the **report_ideal_network** command and the ideal transition *value* option is ignored for those pins. Ideal transition values do not have any effect in ideal clock networks (i.e. non-propagated clock networks).

You can use the **set_ideal_transition** command for pins at lower levels of the design hierarchy.

To list ideal transition values, use the **report_ideal_network** command.

To remove the ideal transition values from a design, use the **remove_ideal_transition** or **reset_design** command.

EXAMPLES

The following example specifies a rise transition of *1.2* and a fall transition of *0.9* for ports named *A*, *B*, and *C*.

```
pt_shell> set_ideal_transition 1.2 -rise {A B C}
pt_shell> set_ideal_transition 0.9 -fall {A B C}
```

SEE ALSO

```
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
report_timing(2)
set_ideal_latency(2)
set_ideal_network(2)
```

set_input_delay

Defines the arrival time relative to a clock.

SYNTAX

```
string set_input_delay
   [-clock clock_name]
   [-reference_pin pin_port_name]
   [-clock_fall]
   [-level_sensitive]
   [-rise]
   [-fall]
   [-max]
   [-min]
   [-add_delay]
   [-network_latency_included]
   [-source_latency_included]
   delay_value
   port_pin_list
```

Data Types

clock_name	list
pin_port_name	list
delay_value	float
port_pin_list	list

ARGUMENTS

-clock *clock_name*

Specifies the clock to which the specified delay is related. If you use the *-clock_fall* option, you must specify the *-clock clock_name* option. If you do not specify the *-clock* option, the delay is relative to time zero for combinational designs. Without using *-clock*, the input delay is not relative to any clock and will not create constrained paths from the port with repspect to any clock. See the variable *timing_input_port_default_clock* to change this behavior.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you use this option and the propagated clocking is being used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The *-network_latency_included* and *-source_latency_included* options cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where

you are setting the input delay and if the *-clock* option is not used, the command considers all of the clocks.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock. When it is used with *-reference_pin* option, the delay is relative to the falling transition of the reference pin. The default is the rising edge or rising transition of a reference pin.

-level_sensitive

Specifies that the source of the delay is a level-sensitive latch. This allows the tool to derive setup and hold relationship for paths from this port as if it were a level-sensitive latch. If you do not use the *-level_sensitive* option, the input delay is treated as if it were a path from a flip-flop.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* option, rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* option, rising and falling delays are assumed to be equal.

-max

Specifies that *delay_value* refers to the longest path. If you do not specify the *-max* or *-min* option, maximum and minimum input delays are assumed to be equal.

-min

Specifies that *delay_value* refers to the shortest path. If you do not specify the *-max* or *-min* option, maximum and minimum input delays are assumed to be equal.

-add_delay

Specifies whether to add delay information to the existing input delay or to overwrite. Use the *-add_delay* option to capture information about multiple paths leading to an input port that are relative to different clocks or clock edges.

For example, **set_input_delay 5.0 -max -rise -clock phi1 {A}** removes all other maximum rise input delay from "A", because the *-add_delay* option is not specified. Other input delays with different clocks or with clock_fall are removed.

In another example, the *-add_delay* option is specified as **set_input_delay 5.0 -max -rise -clock phi1 -add_delay {A}**. If there is an input maximum rise delay for "A" relative to clock "phi1" rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise input delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-network_latency_included

Specifies whether the clock network latency should not be added to the input delay value. If this option is not specified , the clock network latency of the related clock will be added to the input delay value. It has no effect if the clock is propagated or the input delay is not specified with respect to any clock.

-source_latency_included

Specifies whether the clock source latency should not be added to the input delay value. If this option is not specified , the clock source latency of the related clock will be added to the input delay value. It has no effect if the input delay is not specified with respect to any clock.

delay_value

Specifies the path delay. The *delay_value* option must be in units consistent with the technology library used during analysis. The *delay_value* option represents the amount of time that the signal is available after a clock edge. This usually represents a combinational path delay from the clock pin of a register.

port_pin_list

Provides a list of input port or internal pin names in the current design to which *delay_value* is assigned. If you specify more than one object, the objects are enclosed in braces ({}).

DESCRIPTION

Sets input path delay values for the current design. Used with the **set_load** and **set_driving_cell** commands, the input and output delays characterize the operating environment of the current design.

The **set_input_delay** command sets input path delays on input ports relative to a clock edge. Unless specified, input ports are assumed to have zero input delay. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay from a level-sensitive latch, use the *-level_sensitive* option. If the latch is positive-enabled, set the input delay relative to the rising clock edge. If the latch is negative enabled, set the input delay relative to the falling clock edge. If time is borrowed at that latch, add that time borrowed to the path delay from the latch when determining input delay.

The **characterize_context** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

If for a specific clock the maximum delay is less than minimum delay, PrimeTime makes the maximum delay equal to that of the minimum delay.

PrimeTime adds input delay to path delay for paths starting at primary inputs, and to output delay for paths ending at primary outputs.

PrimeTime allows an input port to behave simultaneously as a clock and data port. You can use the **timing_simultaneous_clock_data_port_compatibility** variable to enable or disable the simultaneous behavior of the input port as a clock and data port. When this variable is set to its default of *false*, simultaneous behavior is enabled and you can use the **set_input_delay** command to define the timing requirements for input ports relative to a clock. In this situation, the following applies:

- If you specify the **set_input_delay** command relative to a clock defined at the same port and the port has data sinks, the command is ignored and an error message is issued. There is only one signal coming to port, and it cannot be at the same time data relative to a clock and the clock signal itself.
- If you specify the **set_input_delay** command relative to a clock defined at a different port and the port has data sinks, the input delay is set and controls data edges launched from the port relative to the clock.
- Regardless of the location of the data port, if the clock port does not fanout to data sinks, the input delay on the clock port is ignored and you receive an error message.

  When you set the **timing_simultaneous_clock_data_port_compatibility** variable to *true*, the simultaneous behavior is disabled and the **set_input_delays** command defines the arrival time relative to a clock. In this situation, when an input port has a clock defined on it, PrimeTime considers the port exclusively as a clock port and imposes restriction on the data edges that are launched. PrimeTime also prevents setting input delays relative to another clock.

To control the clock source latency for any clocks defined on an input port, you must use the **set_clock_latency** command.

If a reference pin is not reachable by any given clock, zero arrival time is assumed. If no clock is specified with a reference pin and this reference pin is not reachable by any clock, an unconstrained path is reported. This behavior is changed after version X-2005.06. The new behavior is these invalid constraints are ignored and the path is constrained by whatever it should be as if no such constraints are defined at all. The **check_timing** command generates a warning if the reference pin is not reachable by any active clock, or if multiple clocks reach the reference pin and no clock is specified in the command.

To get a report on the latency calculation using a reference pin, use one of the following commands:

```
report_timing -path_type full_clock
report_timing -path_type full_clock_expanded
```

To list input delays associated with ports, use the **report_port** command. To list input delays of internal pins, use the **report_design** command. To remove input delay values, use the **remove_input_delay** or **reset_design** command.

EXAMPLES

The following example sets an input delay of 2.3 for ports "IN1" and "IN2" on a combinational design. Because the design is combinational, no clock is needed.

```
pt_shell> set_input_delay 2.3 { IN1 IN2 }
```

The following example sets input delay of 1.2 relative to the rising edge of "CLK1" for all input ports in the design.

```
pt_shell> set_input_delay 1.2 -clock CLK1 [all_inputs]
```

The following example sets the input and output delays for the bidirectional port "INOUT1". The input signal arrives at "INOUT1" 2.5 units after the falling edge of "CLK1". The output signal is required at "INOUT1" at 1.4 units before the rising edge of "CLK2".

```
pt_shell> set_input_delay 2.5 -clock CLK1 -clock_fall { INOUT1 }

pt_shell> set_output_delay 1.4 -clock CLK2 { INOUT1 }
```

The following example models the situation where there are three paths to input port "IN1". The first path is relative to the rising edge of "CLK1". The second path is relative to the falling edge of "CLK1". The third path is relative to the falling edge of "CLK2". The *-add_delay* option is used to indicate that new input delay information does not cause old information to be removed.

```
pt_shell> set_input_delay 2.2 -max clock CLK1 -add_delay { IN1 }

pt_shell> set_input_delay 1.7 -max clock CLK1 -clock_fall -add_delay { IN1 }

pt_shell> set_input_delay 4.3 -max clock CLK2 -clock_fall -add_delay { IN1 }
```

In the following example, two different maximum delays and two minimum delays for port "A" are specified using the *-add_delay* option. Since the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If the *-add_delay* option is not used, the new information overwrites the old information.

```
pt_shell> set_input_delay 3.4 -max -clock CLK1 -add_delay { A }
```

```
pt_shell> set_input_delay 5.0 -max -clock CLK1 -add_delay { A }

pt_shell> set_input_delay 1.1 -min -clock CLK1 -add_delay { A }

pt_shell> set_input_delay 1.3 -min -clock CLK1 -add_delay { A }
```

SEE ALSO

all_inputs(2)
characterize_context(2)
create_clock(2)
current_design(2)
remove_input_delay(2)
report_design(2)
report_port(2)
report_timing(2)
check_timing(2)
reset_design(2)
set_driving_cell(2)
set_load(2)
set_output_delay(2)

set_input_noise

Sets a noise bump for a pin or port.

SYNTAX

```
int set_input_noise
   [-add_noise]
   [-above]
   [-below]
   [-low]
   [-high]
   [-height width_value]
   [-width  height_value]
   object_list
```

Data Types

```
width_value       float
height_value      float
object_list       list
```

ARGUMENTS

-add_noise

Specifies whether to add noise information to the existing input noise or to overwrite it. For example, if the **set_input_noise** is used previously, another **set_input_noise** with the *-add_noise* option will sum the noise information with value previously set by the **set_input_noise** command.

-above

Specifies a noise bump for above ground or power rail noise analysis region.

-below

Specifies a noise bump for below ground or power rail noise analysis region.

-low

Specifies a noise bump for ground rail noise.

-high

Specifies a noise bump for power rail noise.

-height *height_value*

Specifies the height of the noise bump. The height is in the voltage units of the library.

-width ***width_value***

Specifies the width of the noise bump. The width is in the time units of the library. 1.0.

object_list

Specifies a list of pins or ports.

DESCRIPTION

Rather than allow PrimeTime SI to calculate propagated noise bumps, you can explicitly specify a noise bump at any port or pin. In this command, you specify the port or pin in the design on which to apply the noise and the characteristics of the noise bump: the type (above/below high/low), the width in library time units, the height in library voltage units (always entered as a positive number).

PrimeTime SI treats this injected noise as propagated noise from the driver of the net connected to the pin. This noise bump overrides any propagated noise that would otherwise be calculated from the driver of the net.

You can specify propagated noise on an output pin rather than a load pin. In that case, the specified noise bump overrides any propagated noise that would otherwise be calculated from the driver. The specified noise bump is propagated through the subnets and attenuated by the parasitic capacitance and resistance specified for the net, until the propagated noise reaches each of load pin on the net.

The **report_noise_calculation** command shows whether input noise information was calculated or annotated by this command.

EXAMPLES

This example specifies a noise bump height of 0.58, width of 1.70 for pin B of cell U1 in the current design.

```
pt_shell> set_input_noise -above -low -width 1.70 -height 0.58\
  [get_pins U1/B]
```

SEE ALSO

report_noise_calculation(2)
remove_input_noise(2)

set_input_transition

Sets a fixed transition time on input or inout ports.

SYNTAX

```
string set_input_transition
   [-rise]
   [-fall]
   [-min]
   [-max]
   [-clock clock_name]
   [-clock_fall]
   transition
   port_list
```

Data Types

clock_name	string
transition	float
port_list	list

ARGUMENTS

-rise

Sets only the rise transition.

-fall

Sets only the fall transition.

-min

Sets transition for minimum conditions.

-max

Sets transition for maximum conditions.

-clock *clock_name*

The input transition is set relative the specified clock. This option is deprecated, and will be ignored.

-clock_fall

Specifies that the transition is relative to the falling edge of the clock. The default is the rising edge. This option is deprecated, and will be ignored.

transition

Port transition value. This is a floating point number greater than or equal to zero.

port_list

A list of input or inout ports.

DESCRIPTION

The **set_input_transition** command specifies a fixed transition time for a list of input or inout ports. This transition time is not affected by the capacitance of the net connected to the port. The transition time calculates delays for nets and cells in the transitive fanout of the port. The port itself has no cell delay.

To display port transition or drive capability information, use the **report_port -drive** command.

There are two other methods of describing port drive capability. The **set_driving_cell** command causes the port to have transition time calculated as if a given library cell was driving the net. The **set_driving_cell** command also has a cell delay equal to the load-dependent portion of the delay driving the net of the library cell. The driving cell approach is accurate for nonlinear delay models even if the capacitance is changed. Another method is to use the **set_drive** command, which models the driver as a linear resistance. The most recent drive command has precedence.

EXAMPLES

This example specifies that ports matching the pattern "DATA_IN*" has a transition time of 0.75 units.

```
pt_shell> set_input_transition 0.75 [get_ports DATA_IN*]
```

SEE ALSO

set_driving_cell(2)
set_drive(2)
report_port(2)
set_clock_transition(2)

set_intrinsic_rc

Assigns intrinsic resistance and capacitance (RC) values to the PG pins of cells, to be used in rail analysis.

SYNTAX

```
status set_intrinsic_rc
   -rc_list intrinsic_rc
   [-lib_cell library_cell]
   [-pg_pin pg_pin]
   object_list
```

Data Types

```
list intrinsic_rc
list object_list
list library_cell
string pg_pin
```

ARGUMENTS

-rc_list *intrinsic_rc*

Defines the list of resistance and capacitance values to be assigned to the specified objects. When this command is invoked before the **open_mw_cel** or **open_block** command, the resistance unit is assumed as Ohm, and the capacitance unit is assumed as Farad. When this command is invoked after the **open_mw_cel** or **open_block** command, the unit for resistance and capacitance will be from the technology library (which is the first library defined in the link_path).

-lib_cell *library_cell*

Specifies a list library master cells on which PG pins will be set with intrinsic RC values. This option has to be used with the *-pg_pin* option.

This option can only be used before the **open_mw_cel** or **open_block** command, and is mutually exclusive with the option *object_list* option.

-pg_pin *pg_pin*

Specifies PG pin names on which to set intrinsic RC values. This option has to be used with the *-lib_cell* option.

object_list

Specifies a list of PG pins on which to set intrinsic RC values. This option is mutually exclusive with the *-lib_cell* option, and can only be used after the **open_mw_cel** or **open_block** command.

DESCRIPTION

The **set_intrinsic_rc** command assigns a list of RC values {R1 C1 R2 C2 ...} to the PG pins. The command returns 1 if successful, 0 if unsuccessful.

This command can be used either before or after the design is loaded using the **open_mw_cel** or **open_block** command. When it is used before the design loading command, the *-lib_cell* and *-pg_pin* options are allowed. When it is used after the design loading command, only the *object_list* option is allowed.

This command is generally used to provide RC information to the PG pins when the library file does not have correct RC values for dynamic rail analysis. Use the command to modify the RC values for the PG pins that were calculated by the **update_currents** command.

For macro cell PG pins, users are allowed to assign one pair of RC vales only, {0.0 C1}, with zero resistance. Otherwise, a warning message will be issued.

When the **set_instrinic_rc** command run is complete, the PG pin's *intrinsic_rc* attribute for the affected PG pins reflects the assigned RC values.

EXAMPLES

The following example sets the RC values (0 Ohm, 1.0 pF) for the vdd pin of the library cell master INV64 before the **open_mw_cel** command. Note the resistance unit is Ohm, and capacitance unit is Farad.

```
pr_shell> set_intrinsic_rc -rc_list {0.0 1.0e-12} -lib_cell INV64 -pg_pin vdd
pr_shell> open_mw_cel -lib my_lib my_design
```

The following example sets the RC values for the vdd pin of the instance U153 after the **open_mw_cel** command. Note the unit of resistance and capacitance is from the main library.

```
pr_shell> open_mw_cel -lib my_lib my_design
pr_shell> set_intrinsic_rc -rc_list {1.0 2.0 3.0 4.0} U153/vdd
```

The following example sets the RC values for all the cell PG pins named vdd2.

```
pr_shell> open_mw_cel -lib my_lib my_design
pr_shell> set_intrinsic_rc -rc_list {1.0 2.0 3.0 4.0} [get_pg_pins */vdd2]
```

SEE ALSO

```
get_attribute(2)
remove_intrinsic_rc(2)
```

set_isolation

Defines the UPF isolation strategy for the power domains in the design.

SYNTAX

```
int set_isolation
   -domain power_domain
   [-isolation_power_net supply_net_name]
   [-isolation_ground_net supply_net_name]
   [-isolation_supply_set supply_set_name]
   [-no_isolation]
   [-elements objects]
   [-clamp_value 0 | 1 | latch | Z]
   [-applies_to input | output | both]
   [-source source_supply_set_name]
   [-sink sink_supply_set_name]
   [-diff_supply_only TRUE | FALSE]
   [-isolation_signal signal]
   [-location self | parent | fanout]
   [-isolation_sense sense]
   [-force_isolation]
   [-name_prefix prefix_string]
   [-name_suffix suffix_string]
   isolation_strategy
```

Data Types

```
power_domain              string
supply_net_name           string
supply_set_name           string
objects                   list
source_supply_set_name    string
sink_supply_set_name      string
prefix_string             string
suffix_string             string
isolation_strategy        string
signal                 string
sense                  string
location                  string
```

ARGUMENTS

-domain *power_domain*

Specifies the power domain name that this UPF isolation strategy is applied to.

-isolation_power_net *supply_net_name*

Specifies the isolation power net for the isolation cells that are created based on this UPF isolation strategy.

-isolation_ground_net ***supply_net_name***

Specifies the isolation ground net for the isolation cells that are created based on this UPF isolation strategy. At least one of the **-isolation_power_net** or **-isolation_ground_net** options should be specified if the **-no_isolation** option is not specified.

-isolation_supply_set ***supply_set_name***

Specifies the supply set whose power and ground functions are to be used as the isolation power and ground nets respectively. This option is mutually exclusive with the **-isolation_power_net** and **-isolation_ground_net** options.

-no_isolation

Specifies that elements under the influence of this **set_isolation** command should not be isolated.

-elements ***objects***

Specifies the objects that this UPF isolation strategy are applied to. The objects can be pins of the root cells of the power domain, ports of the top-level design for top-level power domains.

-clamp_value 0 | 1 | latch | Z

Specifies the clamp value of the isolation cells that should be created based on this strategy. The default of this option is 0.

-applies_to input | output | both

Specifies whether the given **set_isolation** command applies to all inputs or outputs or both kind of ports of the power domain. The default of this option is **output**.

-source ***source_supply_set_name***

Specifies the source supply set that applies on the elements of the strategy. It filters the ports/pins receiving a net that is driven by logic powered by the supply set. This option is mutually exclusive with the **-applies_to** option.

-sink ***sink_supply_set_name***

Specifies the sink supply set that applies on the elements of the strategy. It filters the ports/pins driving a net that fans out to logic powered by the supply set. This option is mutually exclusive with the **-applies_to** option.

-diff_supply_only TRUE | FALSE

Determines the isolation behavior between driver and receiver supply sets. If the driver is powered by the same supply set as a receiver of the port, no isolation cell is introduced into the path. The default for this option is **FALSE**. The **-source**, **-sink**, and **-diff_supply_only** options are mutually exclusive.

-isolation_signal ***signal***

Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *signal* value can be a pin, port, or net.

-location self | parent | fanout

Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy. The default value is *self*.

A value of *self* specifies that the isolation cell is placed in the hierarchy of the port being isolated.

A value of *parent* specifies that isolation cells is added in the parent hierarchy of the isolating port's hierarchy.

A value of *fanout* specifies that isolation cells is added in the fanout of the isolating port.

-isolation_sense *sense*

Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. Excepted values are either *low* or *high*. The default value is *high*.

-force_isolation

Forces isolation.

-name_prefix *prefix_string*

Prepends the specified prefix to the names of generated isolation instances or nets related to implementation of the isolation strategy. This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

-name_suffix *suffix_string*

Appends the specified suffix to the names of generated isolation instances or nets related to implementation of the isolation strategy. This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

isolation_strategy

Specifies the name of the UPF isolation strategy. The name should be unique within the specified power domain.

DESCRIPTION

This command defines the UPF isolation strategy for the ports of the specified power domain.

If **-elements** and **-applies_to** options are not specified, the isolation strategy is applied to all the outputs ports of the power domain.

If neither the **-isolation_supply_set**, **-isolation_power_net**, nor **-isolation_ground_net** option is specified, the isolation power and ground nets are automatically set to the power and ground functions of the **default_isolation** handle of the power domain, for which the strategy is being defined.

PrimeTime uses the **set_isolation** command to build virtual power ground connectivity. The command creates explicit connection for isolation power and ground nets to power and ground pins of isolation cells. If a cell matches multiple isolation rules (**set_isolation** and **set_isolation_control** commands), the last entered **set_isolation** command rule is used to derive the PG connectivity of the cell. Use the *upf_isolation_strategy* cell attribute to check which isolation rule was applied to the cell.

EXAMPLES

The following example shows how to define a UPF isolation strategy for all the output ports of the specified power domain PD1. Power domain PD1 is defined on the shutdown_inst instance.

```
pt_shell> set_isolation isolation_1 -domain PD1 \
            -isolation_power_net PN1 \
```

```
            -isolation_ground_net GN1
```

The following example shows how to define a UPF isolation strategy for specific ports of the specified power domain.

```
pt_shell> set_isolation isolation_2 -domain PD1 \
            -isolation_power_net PN1 \
            -isolation_ground_net GN1 \
            -elements shutdowm_inst/special_port
```

The following example shows how to find all isolation cells in the design and the strategy applied to each cell:

```
pt_shell> foreach_in_collection c [sort_collection [get_cells -hierarchical *
            -filter "upf_isolation_strategy != {}"] full_name] {
               echo [format {  Cell  %-8s  isolation %s}
                  [get_attribute $c full_name] [get_attribute $c upf_isolation_strategy]]
            }
```

The following example shows how to define a UPF isolation strategy using a supply set:

```
pt_shell> set_isolation isolation_2 -domain PD1 \
            -isolation_supply_set iso_supply_set \
            -elements shutdown_inst/special_port
```

The following example shows how to define a UPF isolation strategy without using a supply set or supply nets.

```
pt_shell> set_isolation isolation_2 -domain PD1 \
            -elements shutdown_inst/special_port
```

The following example shows how to define a UPF isolation strategy using the **-source** and **-sink** options.

```
pt_shell> set_isolation isolation_2 -domain PD1 \
            -isolation_supply_set iso_supply_set \
            -source PD_primary_set \
            -sink sset
```

The following example shows how to define a UPF isolation strategy using the **-diff_supply_only** option.

```
pt_shell> set_isolation isolation_2 -domain PD1 \
            -isolation_supply_set iso_supply_set \
            -source PD_primary_set \
            -diff_supply_only TRUE
```

SEE ALSO

create_power_domain(2)
create_supply_net(2)
create_supply_set(2)
set_isolation_control(2)

set_isolation_control

Provides additional options needed for creating isolation cells. This command is needed with most **set_isolation** commands.

SYNTAX

```
status set_isolation_control
   -domain domain
   -isolation_signal signal
   [-location self | parent | fanout]
   [-isolation_sense sense]
   isolation_name
```

Data Types

domain	string
signal	string
sense	string
isolation_name	string

ARGUMENTS

-domain *domain*

Specifies the power domain name to which this UPF isolation strategy is applied.

This option is required.

-isolation_signal *signal*

Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *signal* value can be a pin, port, or net.

This option is required.

-location self | parent | fanout

Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy. The default value is *self*.

A value of *self* specifies that the isolation cell is placed in the hierarchy of the port being isolated.

A value of *parent* specifies that isolation cells is added in the parent hierarchy of the isolating port's hierarchy.

A value of *fanout* specifies that isolation cells is added in the fanout of the isolating port.

-isolation_sense *sense*

Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. Excepted values are either *low* or *high*. The default value is *high*.

isolation_name

Specifies the UPF isolation strategy name. The isolation strategy should already be defined in the specified power domain.

This argument is required.

DESCRIPTION

This command applies to an existing isolation strategy specified by the **set_isolation** command. It provides extra parameters needed for creating isolation cells.

PrimeTime uses additional information about the isolation strategy provided by this command to properly match isolation strategies to individual leaf cells (PrimeTime does not create isolation cells). Isolation strategy is used for deriving PG connectivity from UPF description.

EXAMPLES

The following example shows how to define a UPF isolation strategy for all output ports of the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst, and isolation strategy isolation_1 is already defined

```
pt_shell> set_isolation_control isolation_1 -domain PD1 \
            -location parent \
            -isolation_signal en \
            -isolation_sense high \
```

SEE ALSO

set_isolation(2)

set_latch_loop_breaker

Specifies transparent latch data pins to be used as loop breaker latch data pins when the **timing_enable_through_paths** variable is set to **true**.

SYNTAX

```
status set_latch_loop_breaker
   -pin pin_list
   [-remove]
   [-avoid]
```

Data Types

```
pin_list     list
```

ARGUMENTS

-pin *pin_list*

Specifies data pins of transparent latches to be loop breakers.

-remove

Removes the user-specified setting on pins.

-avoid

Avoids using the specified pins as loop breakers. Requests to avoid using specified pins as loop breaker pins cannot always be honored, especially if there is a loop from the pin through combinational logic to itself.

DESCRIPTION

When sequential loops of transparent latches exist in a design, the tool selects certain latch data pins for special analysis as loop breaker data pins. Reporting paths through these latch data pins is not permitted except by using the **-trace_latch_borrow** option of the **report_timing** command.

Use the **set_latch_loop_breaker** command to guide the tool in selecting data pins as loop breaker data pins. By guiding the tool to select some latch data pins as loop breakers and not others, you can report the paths of interest.

Before using the **set_latch_loop_breaker** command, set the **timing_enable_through_paths** variable to **true**.

EXAMPLES

The following example specifies that the L1/D pin is a loop breaker pin.

```
pt_shell> set_latch_loop_breaker -pin [get_pin L1/D]
```

SEE ALSO

```
get_latch_loop_groups(2)
report_latch_loop_groups(2)
timing_enable_through_paths(3)
```

set_level_shifter_strategy

Sets the type of strategy to use for reporting the signal level mismatches in the design.

SYNTAX

```
status set_level_shifter_strategy
   [-rule strategy]
```

Data Types

```
strategy      string
```

ARGUMENTS

-rule *strategy*

Specifies types of source-sink pairs of mismatching signal levels that the **check_timing -include signal_level** command reports. The following values are allowed:

- **all** (the default) - Reports all voltage level mismatches.
- **low_to_high** - Reports the voltage level mismatches when a source at a lower voltage drives a sink at a higher voltage.
- **high_to_low** - Reports the voltage level mismatches when a source at a higher voltage drives a sink at a lower voltage.

DESCRIPTION

This command specifies the strategy for reporting nets that need insertion of level-shifters.

EXAMPLES

The following example sets the level-shifter strategy to the **high_to_low** option:

```
prompt> set_level_shifter_strategy -rule high_to_low
```

SEE ALSO

```
check_timing(2)
set_level_shifter_threshold(2)
```

set_level_shifter_threshold

Sets the minimum threshold beyond which the voltage adjustment is required.

SYNTAX

```
integer set_level_shifter_threshold
   -voltage volt
   -percent diff
```

Data Types

```
volt      float
diff      float
```

ARGUMENTS

-voltage *volt*

The absolute difference between the source and sink voltages.

-percent *diff*

The percentage by which the source and sink voltages must differ. The percentage is determined as follows:

```
abs(driver(v)-load(v))/driver(v)*100
```

DESCRIPTION

This command specifies the minimum threshold value for the voltage difference between a source and a sink. Voltage differences above the minimum threshold are reported by **check_timing -include signal_level**. The threshold can be specified as the absolute difference in voltages or a percentage difference or both.

When both **-voltage** and the **-percent** options are both specified: Since physical tools insert level shifters when either absolute or relative threshold is exceeded the **check_timing -include signal_level** only skips signal level mismatches that are smaller than both the absolute and the relative thresholds.

The default threshold voltage and percentage is zero. Therfore you typically must specify both absolute and relative thresholds to suppress reporting of below-threshold level mismatches.

EXAMPLES

The following example sets the level shifter threshold value to the absolute difference of 0.1 (and sets large relative threshold):

```
psyn_shell-t> set_level_shifter_threshold -voltage 0.1 -percent 100
```

The following example sets the threshold value 5 percent (and sets large absolute threshold):

```
prompt> set_level_shifter_threshold -percent 5 -voltage 10
```

The following example sets the threshold value to 0.1 difference and 5 percent, so if either value is reached, the signal level mismatch is reported:

```
prompt> set_level_shifter_threshold -voltage 0.1 -percent 5
```

SEE ALSO

check_timing(2)
set_level_shifter_strategy(2)

set_lib_rail_connection

Sets a physical power pin on a library cell.

SYNTAX

```
status set_lib_rail_connection
   -lib_cell_names lib_cells
   -lib_pin_names library_pin_names
   -lib_rail_names power_rail_names
```

Data Types

```
lib_cells            list
library_pin_names    list
power_rail_names     list
```

ARGUMENTS

-lib_cell_names *lib_cells*

Recognizes the specified list of library cells on power rails.

-lib_pin_names *library_pin_names*

Annotates the specified library pin names.

-lib_rail_names *power_rail_names*

This argument is accepted but not currently supported in PrimeTime.

DESCRIPTION

The **set_lib_rail_connection** command associates the name of a power pin with a library cell. When linking physical Verilog to a logic library, pins specified by this command can be present in the Verilog but not in the logic library.

EXAMPLES

The following example shows how to set a physical power pin on a library cell to link with to logic library.

```
pt_shell> set_lib_rail_connection -lib_cell_names typ/INVX1 -lib_pin_names VDD1
```

When subsequently linking physical Verilog, pins named VDD1 on the INVX1 library cell are ignored when linking to the typ logic library. This allows successful linking of physical Verilog to logic libraries in which power pins not specified.

SEE ALSO

read_verilog(2)

set_library_driver_waveform

Sets the driver waveform used to characterize the timing library.

SYNTAX

```
status set_library_driver_waveform
  -type ramp | standard
  [lib_objs]
```

Data Types

```
lib_objs         list
```

ARGUMENTS

-type ramp | standard

Specifies the type of the waveforms:

- **ramp** - Linear waveforms.
- **standard** - Waveforms commonly known as Synopsys predriver.

lib_objs

List of libraries or library cells on which the driver waveform applies.

DESCRIPTION

The data tables in the timing libraries are indexed by slew. The library characterization process could use the standard Synopsys predriver waveform or a simple ramp waveform to characterize the library. The library data reflects the type of waveform used for characterization. The shape of the waveform generally has only a small effect on the timing results. However, for some forms of advanced analysis, the waveform shape can become significant:

- For advanced crosstalk delay analysis using a CCS noise library, using the exact waveform shape in PrimeTime SI is necessary to get the best possible accuracy.
- When the **write_spice_deck** command writes out a SPICE deck for a given path or arc, each input to the cell should use the same waveform that was used to characterize the library, to get the best possible correlation between PrimeTime SI and SPICE simulation results.

This command specifies the waveform shape for the library objects. This command allows two type of waveforms:

ramp and *standard* (Synopsys predriver waveform). If the library object is not provided the waveform is applied to the whole design. When waveform specified on a library cell it gets higher priority than the waveform set on library and the waveform set on library gets higher priority than the design. This command triggers the **update_timing** command.

EXAMPLES

This example specifies how to set ramp for the whole design. When the **write_spice_deck** command generates the SPICE deck, it uses ramp for voltage sources if no other waveform is specified on the library or library cell.

```
pt_shell> set_library_driver_waveform  -type ramp
```

In the following example the *libOld* is characterized with ramp and the *libNew* library is characterized with standard waveforms. It shows how to set these waveforms after the libraries read to PrimeTime. The *lib_pin* attribute of the *driver_waveform_fall/driver_waveform_rise* could be used to get the name of the characterization waveforms.

```
pt_shell> set_library_driver_waveform -type ramp [get_lib libOld]
pt_shell> set_library_driver_waveform -type standard [get_lib libNew]
pt_shell> get_attr [get_lib_pin libOld/INV/A] driver_waveform_fall
pt_shell> get_attr [get_lib_pin libOld/INV/A] driver_waveform_rise
```

SEE ALSO

```
update_timing(2)
write_spice_deck(2)
delay_calc_waveform_analysis_mode(3)
```

set_load

Sets the capacitance on the specified ports and nets in the current design.

SYNTAX

```
status set_load
  [-min]
  [-max]
  [-rise]
  [-fall]
  [-pin_load]
  [-wire_load]
  [-subtract_pin_load]
  [-allow_negative_cap]
  capacitance
  objects
```

Data Types

```
capacitance   float
objects       list
```

ARGUMENTS

-min

Indicates that the *capacitance* is the minimum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

-max

Indicates that the *capacitance* is the maximum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

-rise

Indicates that the *capacitance* is the rise capacitance. This option can be used in combination with the **-min** or **-max** option. Use this option only with ports. An error message is generated if the *objects* list contains nets.

-fall

Indicates that the *capacitance* is the fall capacitance. This option can be used in combination with the **-min** or **-max** option. Use this option only with ports. An error message is generated if the *objects* list contains nets.

-pin_load

Indicates that the specified *capacitance* is a pin capacitance. Pin capacitance is not subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the **-pin_load** or **-wire_load** option or specify both options, the **-pin_load** option is the default.

-wire_load

Indicates that the specified *capacitance* is a wire capacitance. Pin capacitance is subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the **-pin_load** or **-wire_load** option or specify both options, the **-pin_load** option is the default.

-subtract_pin_load

Indicates that the current pin capacitances of the specified net are to be subtracted from *capacitance* before the net capacitance value is set. Any resulting negative net capacitance values are set to zero. Use this option only if *capacitance* includes pin capacitances.

-allow_negative_cap

Allows the command allowed to set a negative capacitance. Use this option only with the **-subtract_pin_load** option. After subtracting the pin capacitance, the resulting capacitance can become negative.

capacitance

Specifies the capacitance value to be set on the ports and nets in *objects*. It is in the units of the logic library. The capacitance value must be greater than or equal to zero. Any negative value is set to zero.

objects

Specifies a list of ports and nets in the current design, on which the *capacitance* is to be set. If you specify a name (for example, *net_name*) instead of a real object_list (for example, by using the **get_port** command with the *port_name* option or the **get_net** command with the *net_name* option), the tool first searches the list of all ports and then searches the list of all nets for a match.

DESCRIPTION

This command sets the capacitance to a specified value on specified ports and nets in the current design. If the current design is hierarchical, you must link it using the **link** command.

Depending on the options used, the **set_load** command sets these attributes on the specified objects:

```
wire_capacitance_max
wire_capacitance_min
pin_capacitance_max
pin_capacitance_min
```

If this command is run without any options, the **pin_capacitance_max** attribute is set; in min-max mode, the **pin_capacitance_min** attribute is also set.

For ports, if the **set_load** command is issued with only the **-wire_load** option, the **wire_capacitance_max** attribute is set on the specified ports; in min-max mode, the **wire_capacitance_min** attribute is also set. However, the value is actually counted as part of the total wire capacitance and not as part of the pin/port capacitance.

In min-max mode, using either the **-min** or **-max** option sets only the ***_min** or ***_max** attributes, respectively.

If the **-rise** option is specified, the **pin_capacitance_max_rise** and **pin_capacitance_min_rise** attributes

are set. You can use this option in conjunction with either the **-min** or **-max** option to set only one of the two attributes. The same conditions apply for the **-fall** option and the **pin_capacitance_*_fall** attributes.

The total capacitance on a net is the sum of all pin capacitances, port capacitances, and wire capacitances associated with that net. The specified *capacitance* overrides both the internally-estimated net capacitance value and any net capacitance set by the **read_parasitics** command.

You can also use the **set_load** command for nets at lower levels of the design hierarchy. These nets are specified in the "BLOCK1/BLOCK2/NET_NAME" format. The tool treats capacitances in such a way that timing on a subblock should be the same as timing on the higher-level design, since pin/wire caps on ports represent a part of the higher-level design.

To view capacitance values on ports and nets, use the **report_port** and **report_net** commands, respectively. To remove a capacitance value, use the **remove_capacitance** command; you can remove annotated capacitances from the full design by using the **reset_design** command.

EXAMPLES

The following example sets a capacitance of 2 units on the xyz port.

```
pt_shell> set_load 2 xyz
```

The following example sets a capacitance of 2.5 units (the estimated wire capacitance) plus the capacitance of three inverter input pins from the tech_lib library named on all output ports. It uses the user-defined **pt_shell** variable **port_capacitance** to store this capacitance value.

```
pt-shell> set pin_cap \
          [get_attribute [get_lib_pins tech_lib/IV/A] pin_capacitance]
2.0
pt_shell> set port_capacitance [expr 2.5 + (3 * $pin_cap)]
8.5
pt_shell> set_load $port_capacitance [all_outputs]
1
```

The following example uses the **get_attribute** command to get the value of the **pin_capacitance** attribute on the Z pin of IV from the tech_lib library and sets that value on the input_1 and input_2 ports.

```
pt_shell> set_load \
          [get_attribute [get_lib_pins tech_lib/IV/Z] pin_capacitance] \
          [get_ports {input_1 input_2}]
```

The following example sets a capacitance of 3 on the U1/U2/NET3 net. The wire capacitance is set to 3. (Total net capacitance is 3 + the sum of the pin and port capacitances.)

```
pt_shell> set_load 3 U1/U2/NET3
```

The following example sets a total net capacitance (wire capacitance + pin capacitances) of 3 on the U1/U2/NET3 net. If the pin and port capacitances equal 2, the wire capacitance is annotated with 1; if the pin and port capacitances are 3 or greater, the wire capacitance is annotated with 0.

```
pt_shell> set_load -subtract_pin_load 3 U1/U2/NET3
```

The following example sets a wire capacitance of 5 units on the xyz port.

```
pt_shell> set_load -wire_load 5 xyz
```

The following example removes the back-annotated capacitance on a port and on a net.

```
pt_shell> remove_capacitance [get_ports the_answer]
pt_shell> remove_capacitance [get_nets net12]
```

SEE ALSO

all_outputs(2)
current_design(2)
remove_capacitance(2)
report_net(2)
report_port(2)
reset_design(2)
set_drive(2)
set_wire_load_mode(2)

set_max_area

Sets the **max_area** attribute on the current design to a specified value.

SYNTAX

```
int set_max_area
   area_value
```

Data Types

```
area_value      float
```

ARGUMENTS

area_value

Specifies the value to which the *max_area* attribute is to be set. The value must be greater than or equal to 0 (>= 0). The units of the *area_value* option must be consistent with the units in the technology library used during optimization.

DESCRIPTION

The **set_max_area** command sets the *max_area* attribute on the current design to the specified *area_value*. This attribute represents the target area of the design.

Use the **remove_max_area** or **reset_design** command to remove the **max_area** attribute on the current design.

Use the **report_constraint** command to show area cost of the design.

EXAMPLES

The following example sets the target area for the current design to 250.

```
pt_shell> set_max_area 250.0
1
pt_shell> get_attribute [current_design] max_area
250.000000
```

SEE ALSO

```
current_design(2)
get_attribute(2)
remove_max_area(2)
report_constraint(2)
reset_design(2)
```

set_max_capacitance

Sets maximum capacitance for pins, ports, clocks or designs.

SYNTAX

```
string set_max_capacitance
   capacitance_value
   [-clock_path]
   [-data_path]
   [-rise]
   [-fall]
   [-force]
   object_list
```

Data Types

capacitance_value	float
object_list	list

ARGUMENTS

-clock_path

Specifies all pins that are in the network of the specific clocks in the *object_list*.

-data_path

Specifies all pins that are in the data paths launched by the specific clocks in the *object_list*.

-rise

Specifies rising capacitance for all selected pins.

-fall

Specifies falling capacitance for all selected pins.

-force

Force the limit to be set to user-defined value, ignoring more restrictive constraints when they exist.

capacitance_value

Capacitance limit (Value >= 0). This is the maximum total capacitance (pin plus wire capacitance) in library capacitance units.

object_list

Provides a list of pins, ports, clocks or designs on which to set maximum capacitance.

DESCRIPTION

Specifies a maximum capacitance on pins, ports or design. If maximum capacitance is set on a pin or port, the net connected to that pin or port is expected to have a total capacitance less than the specified with the *capacitance_value* option. If specified on a design, the default maximum capacitance for that design is set. Library cell pins also can have a *max_capacitance* value specified.

If maximum capacitance is set on a clock, the maximum capacitance is applied to all pins in this specified clock domain. Within a clock domain, you can optionally restrict the constraint further to clock paths only or data paths only, and to rising or falling capacitance only. Note that the *clock_path*, *data_path*, *rise*, and *fall* options can only be used when the max capacitance limit is set on a clock and does not apply to design, pin, or port.

The most restrictive of the design limit, pin, clock, library, or port limit is used.

The **report_constraint -max_capacitance** command shows maximum capacitance constraint evaluations. The **report_port -design_rule** command shows port maximum capacitance limits. The **report_design** command shows the default maximum capacitance setting for the current design.

To remove maximum capacitance limits from designs or ports, use the **remove_max_capacitance** command.

EXAMPLES

The following example sets a maximum capacitance limit of 2.0 units on ports "OUT*".

```
pt_shell> set_max_capacitance 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum capacitance limit of 5.0 units on the current design.

```
pt_shell> set_max_capacitance 5.0 [current_design]
```

The following example sets a maximum capacitance limit of 0.8 units on all pins of the clock network of CLK.

```
pt_shell> set_max_capacitance 0.8 [get_clocks CLK] -clock_path
```

The following example sets a maximum capacitance limit of 0.7 units on all pins of the data path of CLK.

```
pt_shell> set_max_capacitance 0.7 [get_clocks CLK] -data_path
```

SEE ALSO

```
current_design(2)
remove_max_capacitance(2)
report_constraint(2)
report_design(2)
report_port(2)
```

```
get_ports(2)
set_min_capacitance(2)
```

set_max_delay

Specifies a maximum delay for timing paths.

SYNTAX

```
status set_max_delay
   [-rise]
   [-fall]
   [-reset_path]
   [-ignore_clock_latency]
   [-from from_list]
    [-rise_from rise_from_list]
    [-fall_from fall_from_list]
   [-to to_list]
    [-rise_to rise_to_list]
    [-fall_to fall_to_list]
   [-through through_list]
    [-rise_through rise_through_list]
    [-fall_through fall_through_list]
   [-comment comment_string]
   delay_value
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
comment_string	string
delay_value	float

ARGUMENTS

-rise

Indicates that only rising path delays are to be constrained. If neither the **-rise** nor **-fall** option is specified, both rising and falling delays are constrained.

-fall

Indicates that only falling path delays are to be constrained. If neither the **-rise** nor **-fall** option is specified, both rising and falling delays are constrained.

-reset_path

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with the **-to** option only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with the **-from** and **-to** options, only the paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the **reset_path** command with similar arguments before issuing the **set_max_delay** command.

-ignore_clock_latency

Indicates that the launch and capture clock latencies is to be ignored when computing slack on the specified paths. Note that these paths will be considered as clock-less, and therefore, reporting these paths by their respective ignored clocks will result in unconstrained paths. To report these paths with **ignore_clock_latency**, the user may specify the launched and/or captured devices as options in the **report_timing** command. Additionally, PBA reporting will conservatively use GBA crosstalk delta delays if SI is enabled.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-through ***through_list***

Specifies a list of pins, ports, cells, and nets through which the paths must pass for maximum delay definition. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit the **-through** option, all timing paths specified using the **-from** and **-to** options are affected.

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects.

-comment ***comment_string***

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

delay_value

Specifies a floating point number that represents the required maximum delay value for specified paths. The *delay_value* option must have the same units as the logic library used during analysis. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output delay specified, that delay is added into the path delay.

DESCRIPTION

This command specifies a required maximum delay for timing paths in the current design. The path length for any startpoint in the *from_list* to any endpoint in the *to_list* must be less than the *delay_value*.

The value of a *max_rise_delay* attribute cannot be less than that of a *min_rise_delay* attribute on the same path (and similarly for fall attributes). If this condition occurs, the older attribute is removed.

Individual maximum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, see the **create_clock**, **set_input_delay**, and **set_output_delay** man pages.

The **set_max_delay** command is a point-to-point timing exception command. For example, the command overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_min_delay**, and **set_false_path**. A **set_max_delay** or **set_min_delay** command overrides a **set_multicycle_path** command.

Applying a set_max_delay either using the *from*/**-rise_from**/**-fall_from** options on invalid timing start points or using *to*/**-rise_to**/**-fall_to** option on invalid timing endpoints will cause timing path segmentation. PrimeTime will issue UITE-217 warning under such scenarios. If this happens on a clock path, it will stop the propagation of clock at the point where the set_max_delay was applied.

The more general commands apply to more than one path. For example, either the **-from** or **-to** option is used (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following list of commands is arranged in order, from the highest to the lowest precedence (more specific to more general):

1. set_max_delay -from *pin* -to *pin*
2. set_max_delay -from *pin* -to *clock*
3. set_max_delay -from *clock* -to *pin*
4. set_max_delay -from *pin*
5. set_max_delay -to *pin*
6. set_max_delay -from *clock* -to *clock*
7. set_max_delay -from *clock*
8. set_max_delay -to *clock*

To list the *max_delay*, *min_delay*, *multicycle_path*, and *false_path* information for the design, use the **report_exceptions** command. To list the defined path groups, use the **report_path_group** command.

To remove information set by the **set_max_delay** command, use the **reset_path** or **reset_design** command.

EXAMPLES

The following command specifies that any delay path to port "Y" must be less than 10.0 units.

```
pt_shell> set_max_delay 10.0 -to {Y}
```

The following command specifies that all paths from ff1a or ff1b to ff2e must have delays less than 15.0 units.

```
pt_shell> set_max_delay 15.0 -from {ff1a ff1b} -to {ff2e}
```

The following example specifies that all paths to endpoints clocked by PHI2 must have delays less than 8.5 units.

```
pt_shell> set_max_delay 8.5 -to [get_clocks PHI2]
```

The following example sets a requirement that all paths leading to ports named "busA[*]" must have delays less than 5.0.

```
pt_shell> set_max_delay 5.0 -to "busA[*]"
```

The following example specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} must have delays less than 8.0 units.

```
pt_shell> set_max_delay 8.0 -from ff1/CP -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays less than 8.0 units.

```
pt_shell> set_max_delay 8.0 -from ff1/CP -rise_through {U1/Z U2/Z} \
          -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

create_clock(2)
current_design(2)

```
group_path(2)
report_constraint(2)
report_path_group(2)
reset_design(2)
reset_path(2)
set_false_path(2)
set_input_delay(2)
set_min_delay(2)
set_multicycle_path(2)
set_output_delay(2)
```

set_max_fanout

Sets maximum fanout for input ports or designs.

SYNTAX

```
string set_max_fanout
    fanout_value
   object_list
```

Data Types

```
fanout_value     float
object_list      list
```

ARGUMENTS

fanout_value

Fanout limit (Value >= 0). This is the maximum fanout load in library fanout units.

object_list

Provides a list of input ports or designs on which to set maximum fanout.

DESCRIPTION

Specifies a maximum fanout on input ports or designs. If maximum fanout is set on a port, the net connected to that port is expected to have the **fanout_load** attribute less than the specified **fanout_value** attribute. Fanout load for a net is the sum of the **fanout_load** attributes for all input pins on the net. If specified on a design, the default maximum fanout for that design is set. Library cell pins may also have the **max_fanout** value specified. The most restrictive of the design limit and the pin or port limit will be used.

The **report_constraint -max_fanout** command shows maximum fanout constraint evaluations. The **report_port -design_rule** command shows port maximum fanout limits. The **report_design** command shows the default maximum fanout setting for the current design.

To remove maximum fanout limits from designs or ports, use the **remove_max_fanout** command.

EXAMPLES

The following example sets a maximum fanout limit of 2.0 units on ports "IN*".

```
pt_shell> set_max_fanout 2.0 [ge_ports "IN*"]
```

The following example sets the default maximum fanout limit of 5.0 units on the current design.

```
pt_shell> set_max_fanout 5.0 [current_design]
```

SEE ALSO

current_design(2)
remove_max_fanout(2)
report_constraint(2)
report_design(2)
report_port(2)
get_ports(2)
set_fanout_load(2)
set_min_fanout(2)

set_max_time_borrow

Limits time borrowing for latches.

SYNTAX

```
status set_max_time_borrow
   value
   object_list
```

Data Types

```
value          float
object_list    list
```

ARGUMENTS

value

Specifies the value to which the **max_time_borrow** attribute is set. Specifies the limit of time borrowing on the latches specified in the *object_list* option. The *delay_value* option must be between zero and the default maximum derived from the waveform. By default, the maximum is derived from the ideal clock waveform driving each latch, and is equal to (closing_edge - open_edge). Library setup and data-to-Q propagation times are automatically taken into account. The *delay_value* option is in the same units as those in the logic library used during analysis.

object_list

Specifies a list of objects in the **current_design** command for which time borrowing is to be limited to the *value* option. The objects can be clocks, latch cells, data pins, or clock (enable) pins. If a cell is specified, all enabled pins on that cell are affected.

DESCRIPTION

Sets the **max_time_borrow** attribute to the *value* option to constrain the amount of time borrowing possible for level-sensitive latches. To meet delay targets, the **set_max_time_borrow** command prevents automatic use of all or part of the enabling clock pulse on a latch.

If the **max_time_borrow** attribute is set on both data and clock (enable) pins of a level-sensitive latch, the value set on the data pin takes higher priority.

To get the **max_time_borrow** attributes on the design, use the **get_attribute** or **report_exceptions** command.

To undo the **set_max_time_borrow** command, use the **remove_max_time_borrow** command.

EXAMPLES

The following example restricts time borrowing on latches **latch1a** and **latch2f** to 4.0 units.

```
pt_shell> set_max_time_borrow 4.0 { latch1a latch2f }
```

The following example specifies that no time borrowing will take place on **latch1c**.

```
pt_shell> set_max_time_borrow 0.0 { latch1c }
```

SEE ALSO

```
current_design(2)
get_attribute(2)
remove_max_time_borrow(2)
report_exceptions(2)
```

set_max_transition

Sets maximum transition for pins, ports, clocks, or designs with respect to the main library trip-points.

SYNTAX

```
string set_max_transition
   [-clock_path]
   [-data_path]
   [-rise]
   [-fall]
   [-force]
   transition_value
   object_list
```

Data Types

transition_value	float
object_list	list

ARGUMENTS

-clock_path

Specifies all pins that are in the network of the specific clocks in the object_list.

-data_path

Specifies all pins that are in the data paths launched by the specific clocks in the *object_list* option.

-rise

Specifies rising transition for all selected pins.

-fall

Specifies falling transition for all selected pins.

-force

Force the limit to be set to user-defined value, ignoring more restrictive constraints when they exist.

transition_value

Sets the transition limit (Value >= 0). This is the maximum transition time in library time units.

object_list

Provides a list of pins, ports, clocks or designs on which to set maximum transition.

DESCRIPTION

Specifies a maximum transition on pins, ports or designs. If maximum transition is set on a pin or port, the pin or port is expected to have transition time less than the specified *transition_value* option. If specified on a design, the default maximum transition for that design is set. Library cell pins can also have a *max_transition* value specified. The most restrictive of the design limit and the pin or port limit is used. If user entered limit is the most restrictive limit, it is scaled to that of the pin trip-points during slack computation.

If maximum transition is set on a clock, the maximum transition is applied to all pins in this specified clock domain. Within a clock domain, you can optionally restrict the constraint further to clock paths only or data paths only, and to rising transitions only or falling transitions only.

The **-clock_path**, **-data_path**, **-rise**, and **-fall** options are used only if the list of objects is a clock list, not a pin or port list or design. If a clock list is specified without **-clock_path**, **-data_path**, **-rise** or **-fall**, both clock path and data path, both rise and fall are considered by default.

The **report_constraint -max_transition** command shows maximum transition constraint evaluations. The actual limit used for slack computation is reported. If you use an entered limit for slack computation, the reported limit is different for pins with different trip-points. The minimum of the slack for rise and fall transition is reported. When there are violations of both design/pin/port limit and clock limit in different clock domains, the worst slack is reported.

The **report_port -design_rule** command shows port maximum transition limits. The **report_design** command shows the default maximum transition setting for the current design.

To remove maximum transition limits from designs or ports, use **remove_max_transition**.

EXAMPLES

The following example sets a maximum transition limit of 2.0 units on ports "OUT*".

```
pt_shell> set_max_transition 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum transition limit of 5.0 units on the current design.

```
pt_shell> set_max_transition 5.0 [current_design]
```

The following example sets a maximum transition limit of 2.0 units on all pins in the clock network of CLK.

```
pt_shell> set_max_transition 2.0 [get_clocks CLK] -clock_path
```

SEE ALSO

```
current_design(2)
get_ports(2)
remove_max_transition(2)
report_constraint(2)
report_design(2)
report_port(2)
```

set_message_info

Set some information about diagnostic messages.

SYNTAX

```
string set_message_info -id message_id [-limit max_limit|-stop_on|-stop_ff]

string message_id
integer max_limit
```

ARGUMENTS

-id *message_id*

Information is to be set for the given *message_id*. The message must exist. Although different constraints allow different message types, no constraint allows severe or fatal messages.

-limit *max_limit*

Set the maximum number of occurrences for *message_id*. This is an integer greater than or equal to zero. If you set it to zero, that means the number of occurrences of the message is unlimited. Messages which occur after a limit is reached are automatically suppressed.

-stop_on

Force Tcl error if message is emitted.

-stop_off

Turn off a previous -stop_on directive

DESCRIPTION

The **set_message_info** command sets constraints on diagnostic messages (typically error, warning, and informational messages).

Currently, you can set a upper limit for the number of occurrences of a message. You can set this to zero to indicate that there is no limit. You can retrieve the current limit for a message using the **get_message_info** ***command. When the limit is exceeded, all future occurrences of the message are automatically suppressed. A count of total occurrences (including those suppressed) can be retrieved using get_message_info.***

EXAMPLES

The following example uses **set_message_info** to set a limit on the number of APP-027 messages to 100. When the 101st APP-027 message is about to be issued, you will be warned that the limit has been exceeded, and that all future occurrences will be suppressed.

```
prompt> set_message_info -id APP-027 -limit 100
prompt> do_command
Warning: can't find node U27.1 (APP-027)
Warning: can't find node U27.2 (APP-027)
Warning: can't find node U27.3 (APP-027)
 ...
Warning: can't find node U27.100 (APP-027)
Note - message 'APP-027' limit (100) exceeded.  Remainder will be suppressed.
1
```

SEE ALSO

```
get_message_info(2)
get_message_ids(2)
print_message_info(2)
suppress_message(2)
```

set_min_capacitance

Sets minimum capacitance for ports or designs.

SYNTAX

```
string set_min_capacitance
   capacitance_value
   object_list
```

Data Types

```
capacitance_value     float
object_list           list
```

ARGUMENTS

capacitance_value

Specifies the capacitance limit (Value >= 0). This is the minimum total capacitance (pin plus wire capacitance) in library capacitance units.

object_list

Specifies a list of input or inout ports or designs on which to set minimum capacitance.

DESCRIPTION

Specifies a minimum capacitance on input/inout ports or designs. If minimum capacitance is set on a port, the net connected to that port is expected to have total capacitance greater than the specified *capacitance_value*. If specified on a design, the default minimum capacitance for that design is set. Library cell pins can also have a **min_capacitance** value specified. The most restrictive of the design limit and the pin or port limit is used.

The **report_constraint -min_capacitance** command shows minimum capacitance constraint evaluations. The **report_port -design_rule** shows port minimum capacitance limits. The **report_design** command shows the default minimum capacitance setting for the current design.

To remove minimum capacitance limits from designs or ports, use the **remove_min_capacitance** command.

EXAMPLES

The following example sets a minimum capacitance limit of 0.2 units on ports "IN*".

```
pt_shell> set_min_capacitance 0.2 [get_ports "IN*"]
```

The following example sets the default minimum capacitance limit of 0.1 units on the current design.

```
pt_shell> set_min_capacitance 0.1 [current_design]
```

SEE ALSO

current_design(2)
get_ports(2)
remove_min_capacitance(2)
report_constraint(2)
report_design(2)
report_port(2)

set_min_delay

Specifies a minimum delay for timing paths.

SYNTAX

```
status set_min_delay
   [-rise]
   [-fall]
   [-reset_path]
   [-ignore_clock_latency]
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-comment comment_string]
   delay_value
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list
comment_string	string
delay_value	float

ARGUMENTS

-rise

Indicates that only rising path delays are to be constrained. If neither the **-rise** nor **-fall** option is specified, both rising and falling delays are constrained.

-fall

Indicates that only falling path delays are to be constrained. If neither the **-rise** nor **-fall** option is specified, both rising and falling delays are constrained.

-reset_path

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with only the **-to** option, all paths leading to the specified endpoints are reset. If used with only the **-from** option, all paths leading from the specified startpoints are reset. If used with both the **-from** and **-to** option, only paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the **reset_path** command with similar arguments before issuing the **set_min_delay** command.

-ignore_clock_latency

Indicates that the launch and capture clock latencies is to be ignored when computing slack on the specified paths. Note that these paths will be considered as clock-less, and therefore, reporting these paths by their respective ignored clocks will result in unconstrained paths. To report these paths with **ignore_clock_latency**, the user may specify the launched and/or captured devices as options in the **report_timing** command. Additionally, PBA reporting will conservatively use GBA crosstalk delta delays if SI is enabled.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of he clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-through *through_list*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for maximum delay definition. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port,

a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-comment ***comment_string***

Associates a string description with the command for tracking purposes. When writing out SDC, this option is useful for tagging the methodology or tool that originally synthesized the command. You must begin and end the *comment_string* with double quotation marks (").

delay_value

Specifies a floating point number that represents the required minimum delay value for specified paths. The *delay_value* option must have the same units as the logic library used during analysis. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input external delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output external delay specified, that delay is added into the path delay.

DESCRIPTION

This command specifies a required minimum delay for timing paths in the current design. The path length for any startpoint in *from_list* to any endpoint in *to_list* must be greater than the *delay_value*.

Use the **set_min_delay** command in the following two cases:

1. To set a target minimum delay for output ports in a combinational design.
2. To override the default single cycle timing for paths, where **set_multicycle_path** is not sufficient.

The value of a **min_rise_delay** attribute cannot be greater than that of a **max_rise_delay** attribute for the same path (and similarly for fall attributes). If this condition occurs, the older attribute is removed.

Individual minimum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, see the **create_clock**, **set_input_delay**, and **set_output_delay** man pages.

The **set_min_delay** command is a point-to-point timing exception command; it overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_max_delay**, and **set_false_path**. A **set_max_delay** or **set_min_delay** command overrides a **set_multicycle_path** command.

Applying a set_min_delay either using the *from*/**-rise_from**/**-fall_from** options on invalid timing start points or

using *to*/**-rise_to**/**-fall_to** option on invalid timing endpoints will cause timing path segmentation. PrimeTime will issue UITE-217 warning under such scenarios. If this happens on a clock path, it will stop the propagation of clock at the point where the set_min_delay was applied.

The more general commands apply to more than one path; either the **-from** or **-to** option is used (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following list of commands is arranged in order, from highest to lowest precedence (more specific to more general):

1. set_min_delay -from *pin* -to *pin*
2. set_min_delay -from *pin* -to *clock*
3. set_min_delay -from *clock* -to *pin*
4. set_min_delay -from *pin*
5. set_min_delay -to *pin*
6. set_min_delay -from *clock* -to *clock*
7. set_min_delay -from *clock*
8. set_min_delay -to *clock*

To remove information set by the **set_min_delay** command, use the **reset_path** or **reset_design** command.

EXAMPLES

The following command specifies that any delay path to port "Y" must be greater than 12.5 units.

```
pt_shell> set_min_delay 12.5 -to Y
```

The following command specifies that all paths from A1 and A2 to Z5 must have delays greater than 4.0 units.

```
pt_shell> set_min_delay 4.0 -from {A1 A2} -to Z5
```

The following example specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} must have delays greater than 3.0 units.

```
pt_shell> set_min_delay 3.0 -from ff1/CP -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays greater than 3.0 units.

```
pt_shell> set_min_delay 3.0 -from ff1/CP -rise_through {U1/Z U2/Z} \
          -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

```
current_design(2)
report_constraint(2)
reset_design(2)
set_input_delay(2)
set_output_delay(2)
reset_path(2)
set_false_path(2)
set_multicycle_path(2)
set_max_delay(2)
```

set_min_library

Sets the library to be used for minimum delay analysis

The **set_min_library** command is used to relate a minimum conditions library to a maximum conditions library.

SYNTAX

```
string set_min_library
   [-min_version min_library]
   [-none]
   max_library
```

Data Types

```
  min_library     string
  max_library     string
```

ARGUMENTS

-min_version *min_library*

The library for min analysis. This library is not to be used in the **link_path** command.

-none

Dissociates the *max_library* option from its min library.

max_library

The library for max analysis. This library should be used in the **link_path** command.

DESCRIPTION

The **set_min_library** command creates a max/min relationship between two libraries. After you establish the relationship, the tool uses the *max_library* for maximum delay analysis and the *min_library* for minimum delay analysis.

For each library cell in the *max_library*, the **set_min_library** command searches for a library cell of the same name in the *min_library*. If it is not found, a warning is issued, and there is no max/min relationship for that library cell. If the *min_library* does have the named library cell, the command compares the two library cells to verify that they have the same pins (in the same order, with the same directions), and the same timing arcs. If any of these conditions fails, a warning is issued, and there is no max/min relationship for that library cell.

At least one library cell must match for the command to succeed.

When PrimeTime needs to compute a minimum delay value, if there is a max/min relationship for the library cell, the timing information from the min library is used. Otherwise, the information is taken from the max library.

Both libraries and objects in them can be referenced in commands such as the **set_operating_conditions** and **set_wire_load_model** commands.

Libraries used as the *min_library* in a **set_min_library** command should never be placed in the link path. Only the *max_library*, which is the root of the relationship, is used in the link path. A warning is issued if you use a min library in the link path.

A library in use as a min library can be removed with the **remove_lib** command as long as no environment information (operating conditions, wire load models, and so on) is being used from that library. When you remove a library in this way, any designs using the max/min library is scheduled for a full timing update.

The **list_libs** command shows max/min relationships. In addition, the **report_lib** command indicates when a max library has a relationship to a min library. See the examples or the man pages for these commands.

EXAMPLES

The following example shows a typical usage of **set_min_library**:

```
pt_shell> set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db
Loading db file '/u/libraries/LIB_WC_COM.db'
Loading db file '/u/libraries/LIB_BC_COM.db'
Created max/min library relationship:
  Max: /u/libraries/LIB_WC_COM.db:LIB_WC_COM
  Max: /u/libraries/LIB_BC_COM.db:LIB_BC_COM
1
pt_shell> list_libs
Library Registry:
   m LIB_BC_COM    /u/libraries/LIB_BC_COM.db:LIB_BC_COM
  *M LIB_WC_COM    /u/libraries/LIB_WC_COM.db:LIB_WC_COM

1
pt_shell> set_operating_conditions -max WC_COM -max_library LIB_WC_COM \
          -min BC_COM -min_library LIB_BC_COM
1
pt_shell> report_timing -delay min
 ...
```

SEE ALSO

```
list_libs(2)
remove_lib(2)
report_lib(2)
set_operating_conditions(2)
set_wire_load_model(2)
link_path(3)
```

set_min_pulse_width

Sets a minimum pulse width constraint for specified design objects.

SYNTAX

```
string set_min_pulse_width
   [-low]
   [-high]
   value
   [object_list]
```

Data Types

```
value          float
object_list    list
```

ARGUMENTS

-low

Indicates that the minimum pulse width constraint specified by *value* is to apply only to low clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

-high

Indicates that the minimum pulse width constraint specified by *value* is to apply only to high clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

value

A positive floating point value that specifies the minimum pulse width check to be applied to clock signals.

object_list

Specifies a list of clocks, cells, pins, or ports in the current design, to which the minimum pulse width check is to be applied. If you specify a cell or clock, all pins on the cell or clock are affected. If you do not specify any objects, the minimum pulse width check is applied to the current design.

DESCRIPTION

The **set_min_pulse_width** command specifies a minimum pulse width check to be applied to clock signals in the clock tree or at sequential devices. This value overrides the default values for clock tree minimum pulse width checks. Use this command for sequential clock pin pulse width checks, to add a more restrictive value than the

one specified in library.

A clock pulse width problem occurs if the negation of the clock signal happens too soon after the assertion of the clock signal. Clock pulse width violations can cause the following problems:

- Sequential devices (flip-flops and latches) might not capture data properly.
- In some logic circuits, the entire pulse could disappear.

Two types of minimum pulse width checks are performed:

- Clock Pulse Width Check at Sequential Devices: This check is performed on clock pins of sequential elements. This check verifies that a minimum separation exists between the trailing edge and leading edge of the clock that is clocking the sequential elements. The library defines the constraints, which are environmentally scalable. The **set_min_pulse_width** command overrides the library value. The most restrictive value of pulse width (library-specified or user-asserted) is used. No default value is assumed.
- Clock Tree Pulse Width Check at Logic Circuits: The clock tree pulse width check ensures that the clock pulse is propagated reliably through the logic. This check is performed on the combinational logic circuits through which the clock signals are propagated. No default is assumed for the clock tree pulse width check. The **set_min_pulse_width** command overrides the library value.

Note: If the **timing_enable_pulse_clock_constraints** variable is set to its default of **true**, the constraint set by this command does not apply to pulse clock networks. To apply the constraints to pulse clock networks use the **set_pulse_clock_min_width** command. To constrain pulse clock network using the **set_min_pulse_width** command from UI, set the **timing_enable_pulse_clock_constraints** variable to **false**.

To generate a report of pulse width constraints, use the **report_constraint -min_pulse_width** or **report_min_pulse_width** command. To remove minimum pulse width constraints set by **set_min_pulse_width**, use the **remove_min_pulse_width** command.

The **reset_design** command removes all user-specified attributes from a design, including those set by the **set_min_pulse_width** command.

EXAMPLES

The following example sets a minimum pulse width requirement of 2.0 for both low and high pulses of clock signals.

```
pt_shell> set_min_pulse_width 2.0 [get_clocks CK1]
```

The following example sets a minimum pulse width requirement of 2.5 for the low pulse of the clock signal on the pin U1/Z.

```
pt_shell> set_min_pulse_width -low 2.5 U1/Z
```

SEE ALSO

```
current_design(2)
remove_min_pulse_width(2)
report_constraint(2)
report_min_pulse_width(2)
report_pulse_clock_min_width(2)
reset_design(2)
```

```
set_pulse_clock_min_width(2)
timing_enable_pulse_clock_constraints(3)
```

set_mode

Selects the active mode of cell mode groups or design mode groups

SYNTAX

```
status set_mode
   [-type cell | design]
   [mode_list]
   [instance_list]
```

Data Types

```
mode_list          list
instance_list      list
```

ARGUMENTS

-type cell | design

Indicates the type of mode to be made active. This option has the following mutually-exclusive valid values: **design** and **cell**. The **cell** value specifies that cell modes are to be made active. Cell modes are defined on library cells in the library. The **design** value specifies that design modes are to be made active. Design modes are user-specified using the **define_design_mode_group** and **map_design_mode** commands. If the **-type** value is omitted from the command, then this is equivalent to specifying the **-type cell** value.

mode_list

Specifies a list of modes, each of which is to be made the active mode for its mode group. If the **-type** has a cell value, then the *mode_list* option must contain only cell modes. If the **-type** has a design value, then the mode list must contain only design modes.

instance_list

Specifies a list of instances for which the specified cell modes are made active. This list must only be included with the **-type cell** value.

DESCRIPTION

Selects the active mode for a mode group or for several mode groups and disables modes in the same group as the selected active mode. Mode groups must either have all modes enabled (default setting) or have one of their modes enabled and all others disabled. This command sets either cell modes or design modes depending on the value of the type option.

Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell. When a cell mode is made active for a given instance of the library cell, the cell mode is enabled and all of its timing arcs are enabled for that cell. All other cell modes are automatically disabled. The timing arcs of the disabled cell modes are then automatically disabled. To view what modes have been enabled or disabled use the **report_mode -type** cell. To view what arcs have been disabled, use the **report_disable_timing** command.

In the special case of an arc having multiple cell modes mapped to it, the arc is enabled if any of the modes are enabled.

Cell modes can be made active in three different ways, using the **set_mode -type cell** command, using the **set_mode -type design** command or through evaluation of mode conditions. When conflict arises the following precedence rule is employed: **set_mode** -type cell > **set_mode** -type_design > evaluation of mode conditions

Design modes and design mode groups are defined by you with the **define_design_mode_group** and **map_design_mode** commands. Design modes can be mapped to a set of cell modes or a set of paths. When a design mode of a particular design mode group is made active, all cell modes mapped to the design mode are made active and all paths mapped to the design mode are reset. All other design modes in the design mode group are disabled. All paths associated with these disabled design modes will be set to false path. No action is taken to any cell modes mapped to the disabled design modes.

EXAMPLES

In the following example, the first command defines two design modes, DM1 and DM2. (Since no mode group name was specified, the mode group is named "default".) The second command specifies that for design mode DM1, the READ cell mode is active for the RAM instance Uram1. The third command specifies that for design mode DM2, the WRITE component mode is active for the RAM instance Uram1. The fourth command maps all paths ending at Uram1/D0 to the design mode DM2.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode DM1 READ Uram1
pt_shell> map_design_mode DM2 WRITE Uram1
pt_shell> map_design_mode DM2 -to Uram1/D0
```

The following example selects the design mode DM1 as the active mode for the current design, which in turn causes the cell mode READ to be selected as the active mode for the instance Uram1. Since design mode DM2 is automatically disabled all paths ending at Uram1/D0 become false paths.

```
pt_shell> set_mode -type design DM1
```

The following command directly selects the READ cell mode as the active mode for the RAM instance Uram1.

```
pt_shell> set_mode READ Uram1
```

The following command is equivalent to the previous command.

```
pt_shell> set_mode -type cell READ Uram1
```

SEE ALSO

```
define_design_mode_group(2)
remove_design_mode(2)
```

```
reset_mode(2)
report_mode(2)
map_design_mode(2)
```

set_mtcmos_switch_lib_cell

Sets switching related library values for MTCMOS switch cell masters.

SYNTAX

```
status set_mtcmos_switch_lib_cell
   cell_master_name_list
   [-on_state_resistance res_value |
   -off_state_leakage leakage_value]
   [-input_pg_pin input_pg_pin]
   [-output_pg_pin output_pg_pin]
   -pin_state pin_state_list
   [-pin_state_name pin_state_name]
```

Data Types

```
float res_value
float leakage_value
list cell_master_name_list
list pin_state_list
string input_pg_pin
string output_pg_pin
string pin_state_name
```

ARGUMENTS

-on_state_resistance *res_value*

Specifies an on state linear resistance value to be associated with a MTCMOS switch cell master.

-off_state_leakage *leakage_value*

Specifies an off state leakage value to be associated with a MTCMOS switch cell master.

cell_master_name_list

A list of cell master names on which to apply the value.

-pin_state *pin_state_list*

Specifies a list of pin states to be associated to the annotated value on the MTCMOS switch cell. It should include all input pins in the list.

-input_pg_pin *input_pg_pin*

Specifies the input (or actual) PG pin to which the value is to be associated.

-output_pg_pin ***output_pg_pin***

Specifies the output (or virtual) PG pin to which the value is to be associated.

-pin_state_name *pin_state_name* Specifies a name, pin_state_name to be associated with the pin state described in the pin_state_list.

DESCRIPTION

This command enables you to set an on state resistance, off state leakage or pin state name for a MTCMOS switch cell master which may or may not exist in the library. If the command is invoked before the **open_mw_cel** command, you can use this command to create a new MTCMOS switch cell which does not exist in the library. All MTCMOS switch cell masters created from the command will be put in a library called "rail_sidefile_lib_0". For an existing switch cell (master) you can also overwrite the on state resistance value or off state leakage current value, or create a new pin state. If the command is invoked after **open_mw_cel**, you can only set a pin state name for an existing pin state. To reiterate, the creation of a new MTCMOS library cell (master) or a new pin state is not allowed after **open_mw_cel** is issued. The command **set_annotated_mtcmos_switch_cell** should be used on MTCMOS cell instances in order to set a different on state resistance or off state leakage value. The on state resistance, off state leakage and pin state information are used during rail analysis.

To report library information using previously entered **set_mtcmos_switch_lib_cell** commands, use the command **report_mtcmos_switch_lib_cell.**

EXAMPLES

There is no preexisting MTCMOS cell master named "HEADER" in the Liberty library. The following example will create a new MTCMOS cell master called "HEADER", assign an on state resistance value 100.0 to a pin state, related this to the PG pins and finally name the pin state "low_res". After open_mw_cel, you can get the on state resistance attribute:

```
pr_shell> set_mtcmos_switch_lib_cell HEADER \
            -on_state_resistance 100.0 -pin_state {sw 0} \
            -input_pg_pin VDD -output_pg_pin VVDD \
            -pin_state_name low_res

1
pr_shell> open_mw_cel
1
pr_shell> get_attribute [get_lib_cell */HEADER] on_state_resistance
{low_res "sw 0" VDD VVDD 100.0}
```

A MTCMOS switch cell "FOOTER" has been already defined in the library, an on state resistance is therein defined as 0.52 for pin state {sw1 1 sw2 0} but this has not yet been named. The following example creates an on state resistance for another pin state defined with {sw1 0 sw2 1} and named as "high_res":

```
pr_shell> set_mtcmos_switch_lib_cell FOOTER \
            -on_state_resistance 100.0 -pin_state {sw1 0 sw2 1} \
            -input_pg_pin VDD -output_pg_pin VVDD \
            -pin_state_name high_res
1
pr_shell> open_mw_cel
1
```

```
pr_shell> get_attribute [get_lib_cell */FOOTER] on_state_resistance
{{"" "sw1 1 sw2 0" VDD VVDD 0.52}{high_res "sw1 0 sw2 1" VDD VVDD 100.0}}
```

In the following example, continuing from the last example, a new pin state name "low_res" is defined for an existing pin state {sw1 1 sw2 0} and the command is invoked after open_mw_cel command

```
pr_shell> open_mw_cel -lib design cell
1
pr_shell> set_mtcmos_switch_lib_cell [get_lib_cell */FOOTER] \
            -pin_state {sw1 1 sw2 0} -pin_state_name low_res
1
pr_shell> get_attribute [get_lib_cell */FOOTER] on_state_resistance
{{low_res "sw1 1 sw2 0" VDD VVDD 0.52}{high_res "sw1 0 sw2 1" VDD VVDD 100.0}}
```

SEE ALSO

report_annotated_mtcmos_switch_cell(2)
report_mtcmos_switch_lib_cell(2)
set_annotated_mtcmos_switch_cell(2)

set_multicycle_path

Defines the multicycle path.

SYNTAX

```
status set_multicycle_path
   [-setup]
   [-hold]
   [-rise]
   [-fall]
   [-start]
   [-end]
   [-reset_path]
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-through through_list]
   [-rise_through rise_through_list]
   [-fall_through fall_through_list]
   [-to to_list
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-comment comment_string]
   path_multiplier
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list
comment_string	string
path_multiplier	integer

ARGUMENTS

-setup

Indicates that setup (maximum delay) calculations are to use the specified *path_multiplier*. Note that changing the path_multiplier for setup affects the default hold check as well. If neither **-setup** nor **-hold** is specified, *path_multiplier* is used for setup calculations and 0 is used for hold calculations.

-hold

Indicates that hold (minimum delay) calculations are to use the specified *path_multiplier*. Note that changing the path_multiplier for setup affects the default hold check as well. If neither **-setup** nor **-hold** is specified, *path_multiplier* is used for setup calculations and 0 is used for hold calculations.

-rise

Indicates that only rising path delays are to use *path_multiplier*. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Rise refers to a rising value at the path endpoint.

-fall

Indicates that only falling path delays are to use *path_multiplier*. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Fall refers to a falling value at the path endpoint.

-start

Indicates that the multicycle information is relative to the period of the start clock. The **-start** and **-end** options are needed only for multifrequency designs; otherwise, start and end are equivalent. The start clock is the clock source related to the register or primary input at the path startpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-start** moves the relation backward one cycle of the start clock. A hold multiplier of 1 with **-start** moves the relation forward one cycle of the start clock.

-end

Indicates that the multicycle information is relative to the period of the end clock. The **-start** and **-end** options are needed only for multifrequency designs; otherwise, start and end are equivalent. The end clock is the clock source related to the register or primary output at the path endpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-end** moves the relation forward one cycle of the end clock. A hold multiplier of 1 with **-end** moves the relation backward one cycle of the end clock.

-reset_path

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise/fall setup/hold type is reset. Using this option is equivalent to using the **reset_path** command with similar arguments before issuing the **set_multicycle_path** command.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can

use only one of the **-from**, **-rise_from**, and **-fall_from** options.

-through ***through_list***

Specifies a list of pins, ports, cells and nets through which the multicycle paths must pass. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected.

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a fall transition at the specified objects.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, and **-fall_to** options.

-comment ***comment_string***

Associates a string description with the command for tracking purposes. When writing out SDC, this option is useful for tagging the methodology or tool that originally synthesized the command. You must begin and end the *comment_string* with double quotation marks (").

path_multiplier

An integer that specifies the number of cycles the data path must have for setup or hold, relative to the startpoint or endpoint clock, before data is required at the endpoint. For example, specifying a *path_multiplier* of 2 for setup implies a 2-cycle data path. If you use **-setup**, *path_multiplier* is applied to setup path calculations. If you use **-hold**, *path_multiplier* is applied to hold path calculations. If you do not specify either **-setup** or **-hold**, *path_multiplier* is used for setup and 0 is used for hold. Note that changing the multiplier for setup affects the hold check as well.

DESCRIPTION

This command specifies the number of cycles that the data path must have for setup or hold, so that designated timing paths in the current design have no default setup or hold relations.

PrimeTime applies certain rules to determine single cycle timing relationships for paths between clocked elements; the rules are based on active edges. For flip-flops, a single active edge both launches and captures data. For latches, the open edge launches data and the close edge latches data.

The setup check ensures that the correct data signal is available on destination registers in time to be correctly latched. The default setup behavior, called single-cycle setup, is as follows:

For multifrequency designs, there can be multiple setup relations between two clocks. For every latch edge of the destination clock, the setup check determines the nearest launch edge that precedes each latch edge. The smallest value of (setup_latch_edge - setup_launch_edge) determines the maximum delay requirement for this path.

You can override this default relationship by applying the **set_multicycle_path** or **set_max_delay** command to clocks, pins, ports, or cells. For example, setting the setup path multiplier to 2 with the **set_multicycle_path** command delays the latch edge one clock pulse. Note that changing the setup multiplier also affects the default hold check.

Most often, the setup check moves relative to the end clock. This move changes the data latch time at the path endpoint. In multifrequency designs, use the **set_multicycle_path -setup -start** command to move the data launch time backward. The hold check ensures that data from the source clock edge that follows the setup launch edge is not latched by the setup latch edge, and also ensures that data from the setup launch edge is not latched by the destination clock edge that precedes the setup latch edge.

The hold check is determined relative to each valid setup relationship, after applying multicycle path multipliers. The most restrictive (largest) difference of (hold_latch_edge - hold_launch_edge) is used as a minimum delay requirement for the path.

Important Note: The hold relation is conservative. In some cases you need to override the default hold relation. For multifrequency designs, it is more straightforward to use the **set_min_delay** command to override the default instead of using the **set_multicycle_path -hold** command.

There are separate setup and hold multipliers for rise and fall. In most cases, these are set to the same value. You can use the **-rise** or **-fall** option to apply different values.

The **set_multicycle_path** command is a point-to-point timing exception command. The command can override the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_max_delay**, **set_min_delay**, and **set_false_path**. False path information always takes precedence over multicycle path information. A specific **set_max_delay** or **set_min_delay** command overrides a general **set_multicycle_path** command.

The more general commands apply to more than one path; either **-from** or **-to** (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following lists commands from highest to lowest precedence (more specific to more general):

1. set_multicycle_path -from *pin* -to *pin*
2. set_multicycle_path -from *pin* -to *clock*
3. set_multicycle_path -from *pin*
4. set_multicycle_path -from *clock* -to *pin*
5. set_multicycle_path -to *pin*
6. set_multicycle_path -from *clock* -to *clock*
7. set_multicycle_path -from *clock*
8. set_multicycle_path -to *clock*

To undo a **set_multicycle_path** command, use the **reset_path** command. The **reset_design** command removes all attributes from the design, including those set by the **set_multicycle_path** command.

To disable setup and hold calculations for paths, use the **set_false_path** command. To list the point-to-point exceptions on a design, use the **report_exceptions** command.

EXAMPLES

The following example sets all paths between latch1b and latch2d to 2-cycle paths for setup. Hold is measured at the previous edge of the clock at latch2d.

```
pt_shell> set_multicycle_path 2 -from { latch1b } -to { latch2d }
```

The following example moves the hold check to the preceding edge of the start clock.

```
pt_shell> set_multicycle_path -1 -from { latch1b } -to { latch2d }
```

The following example is a two-phase level-sensitive design, where the designer expects paths from phi1 to phi1 to be zero cycles.

```
pt_shell> set_multicycle_path 0 -from phi1 -to phi1
```

The following example uses **-start** to specify a 3-cycle path relative to the clock at the path startpoint. Clock sources are specified, affecting all sequential elements clocked by that clock, or ports with input or output delay relative to that clock.

```
pt_shell> set_multicycle_path 3 -start -from { clk50mhz } -to { clk10mhz }
```

The following example sets all rising paths to ff12/D to 2 cycles, and falling paths to 1 cycle.

```
pt_shell> set_multicycle_path 2 -rise -to { ff12/D }

pt_shell> set_multicycle_path 1 -fall -to { ff12/D }
```

The following multifrequency example shows a 20ns clock to 10ns clock with multicycle path of 2. Assume a path from ff1 (clocked by CLK2) to ff2 (clocked by CLK1).

```
pt_shell> create_clock -period 20 -waveform {0 10} CLK2
pt_shell> create_clock -period 10 -waveform {0 5} CLK1
pt_shell> set_multicycle_path 2 -setup -from ff1/CP -to ff2/D
```

The single-cycle setup relation is from the CLK1 edge at 0ns to the CLK2 edge at 10ns. With the multicycle path, the setup relation is now 20ns (from CLK1 edge at 0ns to CLK2 edge at 20ns). The hold relations are determined according to that setup relation.

1. Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge. This implies a hold relation of 0ns (from CLK1 edge at 20ns to CLK2 edge at 20ns).
2. Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge. This implies a hold relation of 10ns (from CLK1 edge at 0ns to CLK2 edge at 10ns).

The most restrictive (largest) hold relation is used, so the minimum delay requirement for this path is 10ns. This is a conservative check which might not apply to certain designs. Often you know that the destination register is disabled for the clock edge at 10ns. You can modify this default hold relation with the following to get an 0ns requirement.

```
pt_shell> set_multicycle_path 1 -hold -end -from ff1/CP -to ff2/D
```

However, a simpler approach is to use the following:

```
pt_shell> set_min_delay 0 -from ff1/CP -to ff2/D
```

The following example sets all timing paths from ff1/CP to ff2/D which passes through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} to 2 cycle paths for setup.

```
pt_shell> set_multicycle_path 2 -from ff1/CP -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

current_design(2)
report_exceptions(2)
reset_design(2)
reset_path(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)

set_mw_lib_reference

Sets the reference library for the Milkyway library.

SYNTAX

```
status_value set_mw_lib_reference
   [-mw_reference_library lib_list]
   [-reference_control_file file_name]
   libName
```

Data Types

```
lib_list      string
file_name     string
libName       string
```

ARGUMENTS

-mw_reference_library *lib_list*

Specifies a list of libraries to be set as reference libraries for the Milkyway library.

This argument and **-reference_control_file** are mutually exclusive.

-reference_control_file *file_name*

Specifies a reference control file containing information to set the reference libraries for the Milkyway library.

This argument and **-mw_reference_library** are mutually exclusive.

libName

Specifies the Milkyway library to be worked on.

DESCRIPTION

Sets or changes the reference libraries for the Milkyway library. The Milkyway library being updated must be closed for the command to work correctly. The **-reference_control_file** and **-mw_reference_library** options are mutually exclusive, and at least one of the two must be specified.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the reference libraries with a list:

```
prompt> set_mw_lib_reference -mw_reference_library {./lib/ref1 ./lib/ref2} design
```

SEE ALSO

```
create_mw_lib(2)
set_mw_technology_file(2)
```

set_mw_lib_write_access

Changes the write permission for the open Milkyway library.

SYNTAX

```
status set_mw_lib_write_access
   [-clear]
```

ARGUMENTS

-clear

Resets the write permission of the open Milkyway library to read-only.

DESCRIPTION

This command changes the access permission of the open Milkyway library. When the Milkyway library is initially opened as part of **open_mw_cel**, the access mode is set to read-only. You can change the access permissions to write by executing **set_mw_lib_write_access** without options. When the *-clear* option is specified, the access permissions for the currently opened Milkyway library are set back to read-only.

Some operations currently performed by the tool require write access to the libraries. For instance, the **export_rail_result** command creates an *error cell* in the Milkyway library, and the **extract_supply_parasitics** command creates PARA views in Milkyway.

EXAMPLES

The following example sets the access mode to write for the open Milkyway library:

```
pr_shell> set_mw_lib_write_access
1
```

SEE ALSO

```
export_rail_result(2)
extract_supply_parasitics(2)
open_mw_cel(2)
```

set_noise_derate

Sets noise derating information for the current design. This does not affect aggressor screening and filtering result.

SYNTAX

```
int set_noise_derate
   [-above]
   [-below]
   [-low]
   [-high]
   [-height_offset hoffset]
   [-absolute_height_offset_in_volts abshoffset]
   [-height_factor hfactor]
   [-width_factor wfactor]
   [object_list]
```

Data Types

hoffset	float
abshoffset	float
hfactor	float
wfactor	float
object_list	list

ARGUMENTS

-above

Specifies a derating for above ground or power rails noise analysis region.

-below

Specifies a derating for below ground or power rails noise analysis region.

-low

Specifies a derating for ground rail noise.

-high

Specifies a derating for power rail noise.

-height_offset *hoffset*

Specifies an offset voltage in ratio of Vdd, that is directly added to each aggressor and total noise height of a pin in the design. The value of this offset must be between -1.0 to 1.0. The default for this offset is 0.0.

-absolute_height_offset_in_volts ***abshoffset***

Specifies an offset voltage in absolute voltage units that is directly added to each aggressor and total noise height of a pin in the design. The default for this offset is 0.0.

-height_factor ***hfactor***

Specifies a scale factor that could increase or decrease the calculated height of the noise waveform of each aggressor and the total noise height of a pin in the design. The value for this scale factor must be larger than 0.0. The default for this scale factor is 1.0.

-width_factor ***wfactor***

Specifies a scale factor that could increase or decrease the calculated width of the noise waveform of each aggressor and the total noise width of a pin in the design. The value for this scale factor must be larger than 0.0. The default for this scale factor is 1.0.

object_list

Specifies a list of input pins or output ports.

DESCRIPTION

This command sets the noise derate offset and scale factors

- On the current design if you do not use the *object_list* option
- On the specified pins and ports if you use the *object_list* option

Noise derating offset and scale factors

- Affect the noise height and width values shown in noise reports
- Affect the noise propagated through stages
- Do not affect aggressor filtering or screening

If the offset is not specified for a noise analysis region, 0.0 offset is assumed. The absolute and relative height offsets can be specified together. If scale factors are not specified for a noise analysis region, 1.0 scale factors are assumed.

To show derating offsets and scale factors settings, use the **report_noise_calculation** command.

EXAMPLES

This example specifies a voltage offset of 0.10 and noise height and width scale factors of 0.90 for above the ground rail noise:

```
pt_shell> set_noise_derate -above -low -height_offset 0.10 \
          -height_factor 0.90 -width_factor 0.90
```

To restore the settings to their original values:

```
pt_shell> set_noise_derate -above -low -height_offset 0.0 \
```

```
-height_factor 1.0 -width_factor 1.0
```

SEE ALSO

```
report_noise(2)
report_noise_calculation(2)
set_noise_parameters(2)
```

set_noise_immunity_curve

Specifies the noise immunity curve at an input of a library cell or at an output port of the design.

SYNTAX

```
status set_noise_immunity_curve
   [-above]
   [-below]
   [-low]
   [-high]
   [-height height_value]
   [-width  width_value]
   [-area   area_value]
   object_list
```

Data Types

```
height_value     float
width_value      float
area_value       float
object_list      list
```

ARGUMENTS

-above

Specifies an immunity curve for above ground or power rail noise analysis region.

-below

Specifies an immunity curve for below ground or power rail noise analysis region.

-low

Specifies an immunity curve for ground rail noise. When used with the **-above** or **-below** option, the **-low** option specifies the immunity curve for above-low or below-low rail noise, respectively.

-high

Specifies an immunity curve for power rail noise. When used with the **-above** or **-below** option, the **-high** option specifies the immunity curve for above-high or below-high rail noise, respectively.

-height *height_value*

Specifies the height offset of the noise immunity curve above the horizontal axis, in the voltage units of the library. A smaller value increases the likelihood that violations are reported.

-width *width_value*

Specifies the width offset of the noise immunity curve from the vertical axis, in time units of the library. A smaller value increases the likelihood that violations are reported.

-area ***area_value***

Specifies a relative area under the hyperbolic curve, in units equal to the product of the voltage units and time units of the library.

object_list

Specifies a list of library pins or ports.

DESCRIPTION

Each cell input can tolerate a certain amount of noise without causing a failure at the cell output. This characteristic is called noise immunity. This command specifies noise immunity at library input pins or design output ports to determine whether noise failures occur as well as the amount of noise slack.

Noise immunity in the library can be specified as noise immunity curves, polynomials, tables, or in terms of noise margins that consider only the bump heights at the cell inputs.

This command specifies a noise immunity curve at an input of a library cell or at an output port of the design. Use this command in the absence of library-specified noise immunity characteristics, or to override the library-specified characteristics by replacing them with a noise immunity curve. This command also overrides noise margins that have been annotated using the **set_noise_margin** command,

The noise immunity curve is a hyperbola described by the following equation:

y = C1 + C2 / (x - C3)

where

- y = noise immunity height in library voltage units
- x = bump width in library time units
- C1 = height offset set with the **-height** option
- C2 = area parameter set with the **-area** option
- C3 = width offset set with the **-width** option

A noise violation is reported when bump height is above the hyperbola, or in other words,

bump_height > C1 + C2 / (bump_width - C3)

Specify the three constants C1, C2, and C3 (the **-height**, **-area**, and **-width** options) so that the hyperbolic curve matches the noise immunity points obtained by circuit simulation of the cell containing the pin.

Noise immunity characteristics can vary for different noise bump types, so you can specify four different noise immunity curves associated with each input: below low, above low, below high, and above high. Specify the C1, C2, and C3 constants as positive numbers for all four types of noise bumps.

The **report_noise_calculation** command shows whether noise immunity information was taken from the library or annotated by this command.

EXAMPLES

This example specifies a noise height offset of 0.59 voltage units, a noise width offset of 0.04 time units, and a hyperbolic area constant of 0.0064 voltage-time units for above-the-ground rail noise at the A pin of the IV library cell in the lsi_10k library:

```
pt_shell> set_noise_immunity_curve -above -low -height 0.59 -width 0.04 \
            -area 0.0064 lsi_10k/IV/A
```

A violation is reported when bump_height > 0.59 + 0.0064 / (bump_width - 0.04).

SEE ALSO

remove_noise_immunity_curve(2)
report_noise_calculation(2)
set_noise_margin(2)

set_noise_lib_pin

Sets an equivalent noise library pin for a driver or load.

SYNTAX

```
int set_noise_lib_pin
   pins
   lib_pin
```

Data Types

```
pins        list
lib_pin     list
```

ARGUMENTS

pins

Specifies a collection of pins for which the noise library pin is set.

lib_pin

Specifies the equivalent library pin from which the noise information is used.

DESCRIPTION

This command allows to redefine the noise information for a certain library pin. For output pins, setting an equivalent noise library pin means that the I/V curves of the equivalent noise library pin is used. For input pins, noise immunity information of the equivalent noise library pin is used.

EXAMPLES

This example specifies an equivalence for noise information on two inputs pins of the design to be the same as an input library pin:

```
pt_shell> set_noise_lib_pin [get_pins {cell/pin1 cell/pin2}] lsi_10k/IV/A\
```

SEE ALSO

```
update_noise(2)
report_noise(2)
report_noise_calculation(2)
```

set_noise_margin

Specifies the noise margin for a library pin, port, or pin.

SYNTAX

```
status set_noise_margin
   [-above]
   [-below]
   [-low]
   [-high]
   margin_value
   object_list
```

Data Types

```
margin_value      float
object_list       list
```

ARGUMENTS

-above

Specifies the noise margin for above ground or power rail noise analysis region.

-below

Specifies the noise margin for below ground or power rail noise analysis region.

-low

Specifies the noise margin for ground rail noise.

-high

Specifies the noise margin for power rail noise.

margin_value

Specifies a margin value. The value is the input height in voltage units.

object_list

Specifies a list of library pins, ports, or pins.

DESCRIPTION

Each library cell input can tolerate a certain amount of noise without causing a failure at the cell output. This characteristic is called noise immunity. This command specifies noise immunity at library pins to determine whether noise failures occur as well as the amount of noise slack.

Noise immunity in the library can be specified as noise immunity curves, polynomials, or tables or in terms of noise margins.

This command specifies a noise margin for a library pin, design port, or pin. It can be used in the absence of library-specified noise immunity characteristics, or to override the library-specified characteristics by replacing them with noise margins. Noise immunity curves that have been annotated using the **set_noise_immunity_curve** command overrides noise margins set by this command.

Noise margins consider only the bump heights at the cell inputs. Using height-only noise margins are faster and more conservative than the other methods.

For high, narrow noise bumps, using height-only noise margins is pessimistic because it treats some bumps as noise failures that would otherwise pass with the immunity curve model. However, for wide noise bumps, using noise margins gives the same results as using noise immunity curves.

Noise immunity characteristics can vary for different noise bump types, so there can be four different noise margins associated with each input: below low, above low, below high, and above high. Specify all values as positive numbers for all four types of noise bumps.

In the absence of command-specified or library-specified noise immunity data, the tool calculates the maximum allowable noise bump heights based on DC noise margins of the driver and receiver, as defined in the .lib logic library by the input/output logic-level parameters.

The **report_noise_calculation** command shows whether noise margin information was taken from the library or annotated by this command.

EXAMPLES

This example specifies a noise margin of 4 for above-the-ground rail noise for the A pin of the IV library cell in the lsi_10k library:

```
pt_shell> set_noise_margin -above -low 4 lsi_10k/IV/A
```

SEE ALSO

```
remove_noise_margin(2)
report_noise_calculation(2)
set_noise_immunity_curve(2)
```

set_noise_parameters

Defines the noise analysis parameters for the current design.

SYNTAX

```
status set_noise_parameters
   [-ignore_arrival]
   [-include_beyond_rails]
   [-enable_propagation]
   [-analysis_mode report_at_source | report_at_endpoint]
   [-analysis_type violators | all]
```

ARGUMENTS

-ignore_arrival

Ignores the arrival window information of the aggressors during noise analysis. Therefore, the aggressors are assumed to be always overlapping to maximize the effect of a coupled noise bump.

-include_beyond_rails

Enables the analysis of noise beyond the high and low regions. If you do not use this option, the analysis of noise above the high rail and below the low rail is disabled.

-enable_propagation

Enables noise propagation. Propagated noise on a victim net is caused by noise at an input of the cell that is driving the victim net. PrimeTime SI can calculate propagated noise at a cell output, given the propagation characteristics of the cell, the noise bump at the cell input, and the load on the cell output.

-analysis_mode report_at_source | report_at_endpoint

Specifies one of the following analysis modes:

- **report_at_source** (the default) - Reports violations at the source of violations.
- **report_at_endpoint mode** - Propagates violations through fanout and reports violations at endpoints.

-analysis_type violators | all

Focuses the noise analysis with one of the following analysis types; both analysis types catch all noise violations:

- **violators** (the default) - Evaluates each net in the design for potential noise violations. The tool performs detailed noise analysis on only nets with potential violations. This analysis type is particularly useful if the main purpose of noise analysis is to detect noise violations. This mode can be considerably faster than the **all** analysis type.

- **all** - Performs detailed noise analysis on all nets, regardless of whether they have potential violations.

DESCRIPTION

This command specifies the parameters that are considered during the noise analysis. If you do not use this command, or you use the **reset_noise_parameters** command, the tool follows the default noise analysis behavior -- the tool ignores beyond-the-rail analysis, considers the arrival times of aggressors, and enables noise propagation.

If you do not specify any options with this command, the command has no effect, and the tool follows the default noise analysis behavior.

To report the status of the noise analysis parameters for the current design, use the **report_noise_parameters** command.

To reset the noise analysis parameters to the default settings, use the **reset_noise_parameters** command.

EXAMPLES

In the following example, the noise propagation is enabled while other parameters are not affected.

```
pt_shell> set_noise_parameters -enable_propagation
```

SEE ALSO

```
get_noise_violation_sources(2)
report_noise(2)
report_noise_parameters(2)
report_noise_violation_sources(2)
reset_noise_parameters(2)
update_noise(2)
```

set_operating_conditions

Defines the operating conditions or environmental characteristics for the current design.

SYNTAX

```
status set_operating_conditions
   [-analysis_type single | on_chip_variation]
   [-library lib]
   [condition]
   [-min min_condition]
   [-max max_condition]
   [-min_library min_lib]
   [-max_library max_lib]
   [-object_list objects]
```

Data Types

lib	string
condition	string
min_condition	string
max_condition	string
min_lib	string
max_lib	string
objects	list

ARGUMENTS

-analysis_type single | on_chip_variation

Specifies how the specified operating conditions are used:

- **single** - Specifies that only one operating condition is to be used.
- **on_chip_variation** - Switches the design to min_max mode; specifies that the minimum and maximum operating conditions represent, respectively, the lower and upper bounds of the maximum variation of operating conditions on the chip. All maximum path delays use the maximum operating condition, and all minimum path delays use the minimum operating condition.

-library *lib*

Specifies the library that contains definitions of the environmental characteristics to be used. This is either a library name or a collection object.

condition

Specifies the name of the single operating condition.

-min *min_condition*

Specifies the minimum operating condition used for checking minimum delays and hold violations (see the **report_timing** command with the **-delay min** option). If you use this option, you must also use the **-max** option.

-max ***max_condition***

Specifies the maximum operating condition used for checking maximum delays and setup violations (see the **report_timing** command with the **-delay max** option). If you use this option, you must also use the **-min** option.

-min_library ***min_lib***

Specifies the library that contains the *min_condition*. This is either a library name or a collection object. If you use this option, you must also use the **-min** option.

-max_library ***max_lib***

Specifies the library that contains the *max_condition*. This is either a library name or a collection object. If you use this option, you must also use the **-max** option.

-object_list ***objects***

Applies the operating conditions on the specified the cells or ports. If you do not use this option, the operating conditions are applied to the entire design. This option accepts both leaf cells and hierarchical blocks. You cannot use this option with the **-analysis_type** option. This option is only supported in legacy flows; use the **set_voltage** command instead.

DESCRIPTION

This command defines the operating conditions (or environmental characteristics) for performing timing analysis on the current design. You are in min_max mode if you use the **-min** and **-max** options.

The specified operating conditions must be defined in one of the following ways:

- Use the *min_lib* value for only the *min_condition*
- Use the *max_lib* value for only the *max_condition*
- Use one of the libraries in the **link_path**

The order for a library search is as follows:

- 1. *lib*
- 2. *min_lib* or *max_lib*
- 3. **link_path**

If no operating conditions are specified for a design, the tool uses the default operating condition of the library to which the cell is linked. If the library does not have a default operating conditions, no operating conditions are used.

Operating conditions set by using the **-object_list** option override the operating conditions set on design or higher levels of hierarchy.

To view the operating conditions defined for the current design and to determine the libraries to which the current design are linked, use the **report_design** command.

To view the operating conditions defined in the specified library, use the **report_lib** command.

To view the nondefault operating conditions set on cells, use the **report_cell** command.

To remove operating conditions from the current design, use the **remove_operating_conditions** or **reset_design** command.

When process factor, operating temperature, and operating voltage deviate from their nominal values, the tool uses delay scaling among multiple libraries.

PrimeTime derives cell instance voltages as follows:

- Default supply voltage in library cell definition (voltage_map in Liberty syntax)
- **set_operating_conditions** applied at the design level
- **set_voltage** on a supply net
- **set_voltage** on a PG pin

Since operating condition defines only single voltage value caution should be exercised when setting operating conditions on designs containing multirail cells such as level shifters and power management cells. PrimeTime applies the operating condition voltage value only to the first library power rail, for example, the first nonground voltage_map line in the .lib file. Thus, the order of the voltage_map entries in the .lib file is important in this partial multivoltage flow.

The correct way to completely avoid this issue on multirail cells is to use the proper multivoltage flow, such as the UPF flow, to define all design rails using the **set_voltage** command. To verify voltage values of all rails of a multirail cell use the **report_power_pin_info** command.

An alternative is not to use any **set_operating_conditions** command and thus all power rails default to the library nominal voltage defined by voltage_map. Otherwise, use the design-wide **set_operating_conditions**; however, add the **set_voltage** command on specific PG pins of all multirail cells. Also, Liberty libraries should be created with the first voltage_map to match the nominal voltage nom_voltage.

Note that if the analysis type is set to **single**, only maximum delays from a SDF file are used to annotate timing arcs. Changing from a different analysis type into the single operating condition analysis causes all minimum delays to be overwritten by maximum delay values.

EXAMPLES

The following example shows an operating condition definition, as it appears in the source text of a library.

```
operating_conditions("BCCOM") {
  process : 0.6 ;
  temperature : 20 ;
  voltage : 5.25 ;
  tree_type : "best_case_tree" ;
}
```

The name of this set of operating conditions is BCCOM. The parameters are defined as follows:

- **process** - A floating-point number that represents the characteristics of a semiconductor manufacturing process.
- **temperature** - A floating-point number that represents the temperature of the defined environment.
- **voltage** - A floating-point number that defines the upper boundary of the voltage range in which the defined

environment operates. The lower boundary is always 0.0.

- **tree-type** - The interconnect model for the environment. The tool uses the interconnect model to select a formula for calculating interconnect delays. Three models are available:
  - **best_case_tree** - Uses a lumped RC model.
  - **worst_case_tree** - All loads assume full wire resistance.
  - **balanced_tree** - All loads share the wire resistance evenly.

The following example uses operating condition WCIND found in the my_lib.db library to set the operating conditions to WCIND if the **link_path** is my_lib.db.

```
pt_shell> set_operating_conditions WCIND
```

The following example uses the WCIND operating conditions in the other_lib.db library:

```
pt_shell> set_operating_conditions WCIND -library other_lib.db
```

In the following example, the tool uses

- BCCOM values for the minimum operating conditions when calculating minimum delays
- WCCOM values for the maximum operating conditions when calculating maximum delays

```
pt_shell> set_operating_conditions -min BCCOM -max WCCOM \
          -analysis_type on_chip_variation
```

SEE ALSO

```
create_operating_conditions(2)
define_scaling_lib_group(2)
remove_operating_conditions(2)
report_cell(2)
report_design(2)
report_lib(2)
report_power_pin_info(2)
report_timing(2)
reset_design(2)
set_temperature(2)
set_voltage(2)
default_oc_per_lib(3)
```

set_opposite

Sets two ports to be logically opposite.

SYNTAX

```
status set_opposite
  port1
  port2
```

Data Types

```
port1      string
port2      string
```

ARGUMENTS

port1

Specifies the first input port.

port2

Specifies the second input port.

DESCRIPTION

This command defines two input ports in the current design as logically opposite. This information is used during synthesis to eliminate redundant ports to improve optimization quality.

Logical inconsistencies are checked in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

Use the **reset_design** command to remove this property from a port.

The logical relationship is characterized by the **characterize_context** command and exported to synthesis using the **write_context** command.

EXAMPLES

The following example sets two input ports "A" and "B" to be logically opposite.

```
pt_shell> set_opposite A [get_ports B]
1
```

The following example sets up a contradictory relationship between two ports and generates an error message.

```
pt_shell> set_opposite A B
1

pt_shell> set_equal A B
Error: Can't set opposite ports equal in design 'top': 'A' 'B'. (DDB-23)
0
```

SEE ALSO

```
set_equal(2)
characterize_context(2)
write_context(2)
```

set_output_delay

Sets output path delay values for the current design.

SYNTAX

```
string set_output_delay
   [-clock clock_name]
   [-reference_pin pin_port_name]
   [-clock_fall]
   [-level_sensitive]
   [-rise]
   [-fall]
   [-max]
   [-min]
   [-add_delay]
   [-network_latency_included]
   [-source_latency_included]
   [-group_path group_name]
   delay_value
   port_pin_list
```

Data Types

clock_name	list
pin_port_name	list
group_name	string
delay_value	float
port_pin_list	list

ARGUMENTS

-clock *clock_name*

Specifies the name of a clock to which the specified delay is to be related. If you specify the *-clock_fall* option, you must also specify the *-clock* option.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you use this option, and if propagated clocking is used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The *-network_latency_included* and *-source_latency_included* options cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, and in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where you are setting the input delay, and if the *-clock* option is not used, the command considers all of the

clocks.

-clock_fall

Indicates that the delay is relative to the falling edge of the clock. If you specify the *-clock_fall* with the *-reference_pin* option, the delay is relative to the falling transition of the reference pin. If you specify the *-clock* option, the default is the rising edge or the rising transition of the reference pin. If you specify the *-clock_fall* option, you must also specify the *-clock* option.

-level_sensitive

Indicates that the destination of the delay is a level-sensitive latch. This allows the tool to derive a setup and hold a relationship for paths to this port as if it were a level-sensitive latch. If you do not use the *-level_sensitive* option, the output delay is treated as if it were a path to a flip-flop.

-rise

Indicates that the *delay_value* option refers to a rising transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* options, rising and falling delays are assumed to be equal.

-fall

Indicates that the *delay_value* option refers to a falling transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* options, rising and falling delays are assumed to be equal.

-max

Indicates that the *delay_value* option refers to the longest path. If you do not specify the *-max* or *-min* options, maximum and minimum output delays are assumed to be equal.

-min

Indicates that the *delay_value* option refers to the shortest path. If you do not specify the *-max* or *-min* options, maximum and minimum output delays are assumed to be equal.

-add_delay

Indicates that delay information is added to the existing output delay, instead of overwriting the output delay. By using the *-add_delay* option, you can capture information about multiple paths leading from an output port that are relative to different clocks or clock edges. For example, the **set_output_delay 5.0 -max -rise -clock phi1 {OUT1}** command removes all other maximum rise output delays from **OUT1**, because the *-add_delay* option is not specified. Other output delays with a different clock or with the *-clock_fall* option are removed.

In another example, the *-add_delay* option is specified: **set_output_delay 5.0 -max -rise -clock phi1 -add_delay {Z}**. If there is an output maximum rise delay for Z relative to the clock phi1 rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is a maximum rise output delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-network_latency_included

Specifies whether the clock network latency should not be added to the output delay value. If this option is not specified, the clock network latency of the related clock is added to the output delay value. It has no effect if the clock is propagated or the output delay is not specified with respect to any clock.

-source_latency_included

Specifies whether the clock source latency should not be added to the output delay value. If this option is not specified, the clock source latency of the related clock is added to the output delay value. It has no effect if the output delay is not specified with respect to any clock.

-group_path ***group_name***

Specifies the name of a group into which paths ending at the specified ports or pins are added. If the group does not already exist, one is created. The *-group_path* option of the **set_output_delay** command is equivalent to specifying the **group_path -name group_name -to port_pin_list** command. If you do not specify the *-group_path* option of the **set_output_delay** command, the existing path grouping does not change.

delay_value

Specifies the path delay in units consistent with the technology library used during analysis. The *delay_value* option represents the amount of time before a clock edge that the signal is required. For maximum output delay, this represents a combinational path delay to a register plus the library setup time of that register. For minimum output delay, this value is usually the shortest path delay to a register minus the library hold time.

port_pin_list

Specifies a list of output port or internal pin names in the current design to which the *delay_value* option is assigned. If more than one object is specified, the objects are enclosed in braces ({}).

DESCRIPTION

This command sets output path delay values for the current design. The input and output delays characterize the operating environment of the current design when used with the **set_load** and **set_driving_cell** commands.

The **set_output_delay** command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay to a level-sensitive latch, use the *-level_sensitive* option. If the latch is positive-enabled, set the output delay relative to the rising clock edge; if it is negative-enabled, set the output delay relative to the falling clock edge. If time is borrowed at that latch, subtract that time from the path delay to the latch when determining output delay.

The **characterize_context** command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

PrimeTime adds input delay to path delay for paths starting at primary inputs and to output delay for paths ending at primary outputs.

The *-group_path* option modifies the path grouping. Path grouping affects the maximum delay cost. The worst violator within each group adds to the cost.

If a reference pin is not reachable by any given clock, zero arrival time is assumed. If no clock is specified with a reference pin and this reference pin is not reachable by any clock, an unconstrained path is reported. This behavior is changed after X0506. The new behavior of these invalid constraints are ignored and path is constrained by whatever it should be, as if no such constraints were defined. The **check_timing** command generates a warning if the reference pin is not reachable by any active clock, or if multiple clocks reach the reference pin and no clock is specified in the command.

To get a report on the latency calculation using a reference pin, use the **report_timing -path_type full_clock** or **report_timing -path_type full_clock_expanded** commands.

To list output delays associated with ports, use the **report_port** command. To list output delays of internal pins, use the **report_timing** command. To list the path groups that are defined, use the **report_path_group** command.

To remove output delay values, use the **remove_output_delay** or **reset_design** commands. To modify paths grouped with the *-group_path* option, use the **group_path** command to place the paths in another group or the default group.

EXAMPLES

The following example sets an output delay of 1.7 relative to the rising edge of CLK1 for all output ports in the design.

```
pt_shell> set_output_delay 1.7 -clock CLK1 [ all_outputs ]
```

The following example sets the input and output delays for the bidirectional port INOUT1. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
pt_shell> set_input_delay 2.5 -clock CLK1 -clock_fall { INOUT1 }

pt_shell> set_output_delay 1.4 -clock CLK2 { INOUT1 }
```

The following example models the situation where there are three paths from output port OUT1. One of the paths is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The *-add_delay* option indicates that new output delay information does not cause the removal of old information.

```
pt_shell> set_output_delay 2.2 -max clock CLK1 -add_delay { OUT1 }

pt_shell> set_output_delay 1.7 -max clock CLK1 -clock_fall -add_delay { OUT1 }

pt_shell> set_output_delay 4.3 -max clock CLK2 -clock_fall -add_delay { OUT1 }
```

In the following example, two different maximum delays and two minimum delays for port Z are specified using the *-add_delay* option. Because the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If the *-add_delay* option is not used, the new information overwrites the old information.

```
pt_shell> set_output_delay 3.4 -max -clock CLK1 -add_delay { Z }

pt_shell> set_output_delay 5.0 -max -clock CLK1 -add_delay { Z }

pt_shell> set_output_delay 1.1 -min -clock CLK1 -add_delay { Z }

pt_shell> set_output_delay 1.3 -min -clock CLK1 -add_delay { Z }
```

The following example shows how to use the *-group_path* option to add ports into a named group. Note that without this option, paths to these ports are included in the CLK group.

```
pt_shell> set_output_delay 4.5 -max -clock CLK -group_path busA {busA[*]}
```

SEE ALSO

```
all_outputs(2)
characterize_context(2)
create_clock(2)
```

```
current_design(2)
group_path(2)
remove_output_delay(2)
report_design(2)
report_path_group(2)
report_port(2)
reset_design(2)
report_timing(2)
check_timing(2)
set_driving_cell(2)
set_load(2)
set_max_delay(2)
set_output_delay(2)
```

set_parasitic_corner

Sets a parasitic corner for the timing analysis in the presence of variation-aware parasitics.

SYNTAX

```
status set_parasitic_corner
   -name corner_name
   file_name
```

Data Types

file_name string *corner_name* string

ARGUMENTS

-name *corner_name*

Specifies the name of the parasitic corner to be used from the corner file. A given corner file can contain definitions of a number of parasitic corners and this option specifies which corner definition is to be used to be set as the corner for analysis.

file_name

Specifies the name of the parasitic corner or corner file. The file is supposed to contain the percentage variation values for all the parasitic parameters in the variation-aware parasitics.

DESCRIPTION

The **set_parasitic_corner** command reads parasitic corner information from a disk file and sets it as the corner for analysis in the presence of variation-aware parasitics. The command succeeds only if variation-aware parasitics were previously annotated.

The syntax of the corner parasitic file is very simple. It simply defines various parasitic corners by specifying the VARIATION_RATIOs for all the parameters.

A VARIATION_RATIO signifies the actual amount of variation from the mean (or nominal) for a given parameter. A VARIATION_RATIO is expressed in terms of the percentage variation of a given parameter with respect to its variation coefficient. For example, a given parameter has a variation coefficient of 0.09. The actual amount of variation allowed for this parameter is between -0.27 and 0.27. The VARIATION_RATIO is the multiplier to the variation coefficient that gets us to the actual amount of variation for the parameter. So, by definition, the value of VARIATION_RATIO has to be between -3.0 and +3.0.

Technically, the syntax of a corner file can be expressed as:

```
(CORNER_NAME: <name of the corner>  (PARAM_NAME PARAM_VARIATION_RATIO)*)+
```

The corner file should contain definition for at least one corner. A corner can contain VARIATION_RATIO values for ZERO or more number of parameters. If a parameter is not specified in the corner definition, the VARIATION_RATIO is assumed to be ZERO.

Comments can be written in the file following Tcl style - so, anything after the characters " #" (space and pound) on a line is assumed to be a comment until the end of the line. Only line comments are supported.

An example of the corner file is as follows:

```
CORNER_NAME: CMAX
     M1_W        3.0
     M2_T        3.0
     ILD_c_T    -0.04
CORNER_NAME: RCMAX
 ## my definition of RCMAX
     M1_W        3.0
     M2_T        2.0
     ILD_c_T    -0.22
CORNER_NAME: RCMIN
     M1_W        -3.0
     M2_T        -3.0
     ILD_c_T      0.4
```

This indicates that the CMAX corner parasitics must be inferred where the M1_W parameter is changed by positive (3.0 * variation_coeff of M1_W) from the nominal and so on.

You can issue this command multiple times to change the corner for analysis. Only the last command takes effect. You can remove the parasitic corner setting by using the **remove_parasitic_corner** command.

Note that this command has no effect on the **write_parasitics** command. This command sets the corner only for analysis and does not change the variation-aware parasitics.

The parasitic attributes, such as the **rc_network** and **total_coupling_capacitance** attributes, are all modified to reflect the corner values.

EXAMPLES

The following example reads the parasitics corner file "./corner_file" from disk and uses the settings of the RCMAX corner to set the default parasitic corner on the design with variation-aware parasitics.

```
pt_shell> set_parasitic_corner -name RCMAX ./corner_file
```

SEE ALSO

```
remove_parasitic_corner(2)
read_parasitics(2)
report_annotated_parasitics(2)
```

set_pg_pin_mapping

Sets PG pin mapping for cell masters.

SYNTAX

```
status set_pg_pin_mapping
   cell_master_list
   -from_ground_pin ground_pin_name
   -to_power_pins power_pin_names
```

Data Types

```
list cell_master_list
string ground_pin_name
list power_pin_names
```

ARGUMENTS

cell_master_list

Specifies a list of cell masters on which to apply the PG pin mapping.

-from_ground_pin *ground_pin_name*

Specifies the ground pin where the mapping is applied from.

-to_power_pins *power_pin_names*

Specifies a list of the power pins where the mapping is applied to.

DESCRIPTION

This command defines how a ground pin is mapped to a list of power pins. You need to invoke the command before the **open_mw_cel** command.

The PG pin mapping information is used by the **set_rail_power** command to control how the current is distributed from the power pins to the ground pin. Use the command to overwrite any PG mapping information that is derived from the library.

To report PG pin mapping information that is set by the **set_pg_pin_mapping** command, run the **report_pg_pin_mapping** command and then the **open_mw_cel** command.

EXAMPLES

The cell INV64 has been defined in the library and has power pins {VDD1, VDD2} and ground pins {VSS1, VSS2}. The library-derived PG pin mapping is VDD1 to VSS2, and VDD2 to VSS1. The following commands overwrite the library-derived information by mapping VDD1 to VSS1, and VDD2 to VSS2.

```
pr_shell> set_pg_pin_mapping INV64\
                -from_ground_pin VSS1 -to_power_pins {VDD1} \
pr_shell> set_pg_pin_mapping INV64\
                -from_ground_pin VSS2 -to_power_pins {VDD2} \
pr_shell> open_mw_cel -lib my_design my_cell\
```

The following example maps multiple power pins to a ground pin.

```
pr_shell> set_pg_pin_mapping LVS32 \
                -from_ground_pin VSS1 -to_power_pins {VDD1 VDD2} \
```

SEE ALSO

open_mw_cel(2)
report_pg_pin_mapping(2)
set_rail_power(2)

set_port_attributes

Sets the specified attributes and their value on desired port.

SYNTAX

```
string set_port_attributes
   [-ports port_list]
   [-elements element_list]
   [-applies_to inputs | outputs | both]
   [-driver_supply supply_set_ref]
   [-receiver_supply receiver_supply_set_name]
   [-repeater_supply repeater_supply_set_name]
   [-attribute name_value_pair]
```

Data Types

port_list	list
element_list	list
supply_set_ref	string
receiver_supply_set_name	string
repeater_supply_set_name	string
name_value_pair	string

ARGUMENTS

-ports *port_list*

Specifies the collection of ports on which the attributes must be set. Wildcards are accepted in port names.

-elements *element_list*

Specifies a set of ports on the interface to the specified elements, excluding any supply ports. A design element reference '.' is accepted to designate the current scope.

-applies_to *inputs* | *outputs* | *both*

Specifies whether the given **set_port_attributes** command applies to all inputs or outputs or both kind of ports of the specified element. The argument must be used in conjunction with -elements argument. The default value for this argument is *both*.

-attribute *name_value_pair*

Specifies the name and value of the attribute to be set on the ports specified by the **-ports** option.

This option can be repeated multiple times to specify multiple attributes for the same set of ports in a single command.

-driver_supply *supply_set_ref*

Specifies the supply of the logic driving the port.

-receiver_supply ***receiver_supply_set_name***

Specifies the supply of the logic reading the port.

-repeater_supply ***repeater_supply_set_name***

Specifies the supply set used by a repeater driving the port.

DESCRIPTION

This command sets the attributes and their value on a list of ports specified by the -ports option or on a list of instances specified by -elements option.

Attributes can be set multiple times on the same ports. The latest value prevails.

PrimeTime suite tools only support limited number of attributes. For those un-supported attributes, they will be ignored and a warning message will be issued.

The -receiver_supply and -driver_supply attributes can only be specified on the top level ports and design. A warning will be issued if non top-level design element is specified for -elements option for -driver_supply or -receiver_supply attribute. The attribute on invalid design element is ignored.

The tool will interpret a port attribute specified on a design element as less specific than one specified on a particular port of the design element.

EXAMPLES

The following example sets SS1 as the driver_supply on top-level input port IN1 and sets SS2 as the receiver_supply on top-level output port OUT1.

```
prompt> set_port_attributes -ports {IN1} \
               -driver_supply SS1
prompt> set_port_attributes -ports {OUT1} \
               -receiver_supply SS2
```

The following example sets SS1 as the driver_supply on top-level input ports and sets SS2 as the receiver_supply on top-level output ports using -elements {.} -applies_to. '.' refers to the current scope, which is set to the top-level design in this case.

```
prompt> set_port_attributes -elements {.} \
              -applies_to inputs \
       -driver_supply SS1
prompt> set_port_attributes -elements {.} \
              -applies_to outputs \
              -receiver_supply SS2
```

The following example sets attribute 'snsp_derived' to TRUE on port power_port and ground_port in block Mult.

```
prompt> set_port_attributes -ports {Mult/power_port Mult/ground_port} -attribute iso_sink SS1
1
```

SEE ALSO

`create_supply_set(2)`

set_port_fanout_number

Sets the number of external fanout points on ports.

SYNTAX

```
string set_port_fanout_number
   [-min]
   [-max]
   fanout_number
   port_list
```

Data Types

```
fanout_number      int
port_list          list
```

ARGUMENTS

-min

Specifies the value for minimum condition.

-max

Specifies the value for maximum condition.

fanout_number

Specifies the number of external fanout points (Range: 0 to 100000).

port_list

Specifies a list of ports. Each element in the list is either a collection of ports or a pattern that matches ports on the current design.

DESCRIPTION

Sets the number of external fanout pins for ports in the current design. The number of pins is used (along with wire load models) to calculate capacitance and resistance of nets.

To remove port fanout number values, use the **remove_port_fanout_number** command. To show port fanout number information, use the **report_port -wire_load** command.

EXAMPLES

This example sets the external wire load model for ports OUT1* to 70x70 and specifies an external fanout number of 2.

```
pt_shell> set_wire_load_model 70x70 [get_ports OUT1*]
pt_shell> set_port_fanout_number 2 [get_ports OUT1*]
```

SEE ALSO

```
collections(2)
remove_port_fanout_number(2)
report_port(2)
set_wire_load_model(2)
```

set_power_clock_scaling

Specify clock clock scaling for power analysis.

SYNTAX

```
integer set_power_clock_scaling
   [-period period_value]
   [-ratio ratio_value]
   [clock_objects]
```

Data Types

```
period_value          float
ratio_value           float
clock_objects         list
```

ARGUMENTS

-period *period_value*

Specify the period of clock which is used in simulation.

-ratio *ratio_value*

Specify the clock ratio (period_used_in_VCD_or_SAIF / period_used_in_SDC).

clock_objects

Specify a list of clocks in the design.

DESCRIPTION

Clock frequency used in value change dump (VCD) and Switching Activity Interchange Format (SAIF) files, from logic simulation, for functional verification can differ from the clock frequency used in Synopsys Design Constraints (SDC) file for timing and power analysis. This command tells PrimeTime PX what frequency is used in logic simulation so that PrimeTime PX can scale averaged power numbers properly.

To enable this feature, set the **power_enable_clock_scaling** variable to *true*.

The clock_objects option in this command is a list of clocks in the design that have the specified period or ratio values used in simulation for VCD or SAIF generation. Here the ratio is defined as period-in-simulation / period-in-analysis. The command can be used multiple times, once for each clock. If the same clock is used more than once, a warning message is issued and the clock specified last takes effect. The *-period* and *-ratio* options cannot be

used simultaneously. If no clock object is specified, only the ratio option is allowed and the ratio value is applied to all the nets that do not have an associated simulation clock.

The tool only scales the switching activities from the VCD file or the SAIF file and activities that are propagated or implied. The actual scaling for switching activity and averaged power analysis is performed by the tool during the execution of the update_power command. You can also check the power_base_clock attribute on the nets to identify the clock that controls the activity scaling of the net. The command returns 1 on success and 0 on failure.

To perform repeated scaling, you need to re-specify the scaling factor. The scaling factor is applied w.r.t the original toggle rate . The tool keeps track of the factor by which the toggle rates are already scaled. For example, if a net has an original toggle rate 1.0 and the *-ratio* option of the **set_power_clock_scaling** command is set to 2, the 1st scaling results in the net's toggle rate to 2 (the original toggle rate 1.0 is replaced). To further scale the toggle rates by twice , you need to specify a scaling factor of 4. To come back to the original toggle rates, you can use **set_power_clock_scaling 1** .

The clock scaling is done only for averaged power. This feature cannot be applied to a time-based peak power analysis.

EXAMPLES

A sample script is listed below. In this case, duration logic simulation the SAIF file mac.saif is generated with clock period 24, so you use set_power_clock_scaling period 24 [get_clock clk], and you specify SDC clock in mac.sdc (it could be period 12, for example). Report_power will report the power consumption when the design is operated at SDC clock period 12, rather than logic simulation clock period 24.

```
set power_enable_analysis true
set power_analysis_mode averaged
set search_path         "../src/hdl/gate ../src/lib/snps . "
set link_library        " * core_typ.db"

read_verilog            mac.vg
current_design          mac
link
read_sdc ../src/hdl/gate/mac.sdc
read_parasitics ../src/annotate/mac.spef.gz
read_saif "../sim/mac.saif" -strip_path "tb/macinst"

set_power_clock_scaling -period 24 [get_clock clk]
set power_enable_clock_scaling true
update_power
report_power

quit
```

SEE ALSO

power_enable_clock_scaling(3)

set_power_derate

Sets power derating factors for either the current design or a specified list of instances (cells, library cells, or power groups).

SYNTAX

```
int set_power_derate
   [-switching]
   [-internal]
   [-leakage]
   [-rails rail_list]
   [-groups group_name_list]
   derate_value
   object_list
```

Data Types

```
group_name_list     list
rail_list           list
derate_value        float
object_list         list
```

ARGUMENTS

-switching

Indicates that the *derate_value* specified should be applied to switching power.

-internal

Indicates that the *derate_value* specified should be applied to internal power.

-leakage

Indicates that the *derate_value* specified should be applied to leakage power only. For CCS power library, it should be applied to both gate leakage and subthreshold leakage power.

-groups *group_name_list*

Specifies the power groups to which the *derate_value* is applied. The *derate_value* applies directly to each cell object in the power group.

-rails *rail_list*

Specifies a list of power supply nets on which power derate factor applied. This option is not supported for non-UPF mode. This feature is only valid on the following objects hierarchycal_cell, leaf_cell, Design and, power-groups. Derated values can be reported by *report_power_derate* command.

derate_value

Specifies the power derating factor that is applied to the specified power components as a scalar multiplicative factor.

object_list

Specifies current design or a list of cells or library cells to which the specified power derating factor is applied.

DESCRIPTION

Sets power derating factors on different power components for either the current design, a list of cells, or library cells in the current design. If no object or power group is specified, the power derating factor is set on the current design. The power derating factors are used as a scalar multiplicative factor in power calculation.

Power derating factors affect power analysis results in power reports and power attributes. Power analysis results are multiplied by the derating factors. If power derating factors are not specified, the value of 1.0 is assumed.

If you use the **set_power_derate** command with the **-internal**, **-switching**, or **-leakage** option, the specified power derating factor is only be applied to internal, switching, or leakage power accordingly. If you do not specify the **-internal**, **-switching**, or **-leakage** option, the power derating factor applies to all power components, which includes internal, switching, and leakage power. You can first set a generic power derating factor that applies to all power components, and then refine it by setting specific power derating factors for particular power components on particular design objects.

The *object_list* option can be used to set power specific derating factors on instances (cells or library cells) in the design. On each instance the **-switching**, **-internal**, and **-leakage** options can be used in the same way as previously described to specify exactly how the derating factors should be applied to each instance in the object list.

When applying power derating factors the following priority is used (in decreasing order of precedence):

1. Leaf Cell or Power Group
2. Hierarchical Cell
3. Library Cell
4. Design

Therefore, when power derating factors are being applied for a cell, PrimeTime PX first checks if they have been set for the cell itself (leaf cell). It then checks if it has been set for any of the cells hierarchical parents. If no derate factors are found, it checks for the library cell and after that it uses the factors set globally on the design.

If power derating factors are set multiple times on the same design object for the same power component, the later value overrides the previous value.

Use the **-groups** option to set specific derating factors on all the cell objects in particular power groups. Power group can be user generated or default, such as clock_network, memory, register. They are considered as a collection of leaf and hierarchical cells. When applying power derating factors, the factors are set directly on the cell objects in the collection. There is one exception. Currently in PrimeTime PX, clock network power includes the clock pin power of registers by default. The clock network power derating factors are not applied to registers, but only applied to the clock pin power of the registers when it was included in the clock network power. The power derating factors set on clock network power group are reset with the **reset_power_derate** command on current design or without any option.

In addition, if the **-switching** option is specified and the *object_list* contains any hierarchical cells, all cells within that hierarchical cell and also all lower hierarchical cells have that derating factor set on them.

The **-groups** option can be used to set specific derating factors on all the cell objects in particular power groups. Power group can be user generated or default, such as clock_network, memory, register, etc.. If a derate factor is set on a power group, the specified derating factors are applied to all cell objects in the power group.

Power derating factors affect power values shown in power reports. The intended usage mode of this command is that you first set global or default derates on the design. Then you can use the *object_list* option to set specific derate values for cells or library cells in the design. To set derates globally on a design simply issue the command with no object list specified.

To set derating values back to the default (derating factor of 1.0 for every instance in the design), use the **reset_power_derate** command.

EXAMPLES

The following example sets a power derating factor of value 0.95 on all cells in the design.

```
pt_shell> set_power_derate 0.95
```

The following example sets switching power derate and internal power derate to 0.9.

```
pt_shell> set_power_derate 0.9 -switching -internal
```

The following example sets the power derate to 0.5 for all power components on the default clock network power group.

```
pt_shell> set_power_derate 0.5 -groups clock_network
```

The following example sets a leakage power derating factor of 1.1 on all instances of library cell IV in the library MY_LIB:

```
pt_shell> set_power_derate -leakage 1.1 [get_lib_cells MY_LIB/IV]
```

Assuming that cell, "inv" is a particular instantiation of MY_LIB/IV in the design the following command overwrites the power derating factor for the instance "inv" only:

```
pt_shell> set_power_derate -leakage 1.05 [get_cells inv]
```

The following example illustrates how to set a power derating factor except leakage power on all cells (including sub-hierarchies) within the hierarchy top/H1:

```
pt_shell> set_power_derate -switching -internal 1.05 [get_cells top/H1]
```

SEE ALSO

```
report_power(2)
report_power_calculation(2)
report_power_derate(2)
reset_power_derate(2)
```

set_probes

Install voltage and current probes at specified pg_pins.

SYNTAX

```
string set_probes
   pg_pins
```

Data Types

```
list pg_pins
```

ARGUMENTS

pg_pins

Specifies a list of pg_pins at which voltage and current probes are to be installed. This argument is required.

DESCRIPTION

The **set_probes** command installs voltage and current probes at the specified list of pg_pins. When *-fsdb_file* option is used in **calculate_rail_voltage** command, both voltage and current waveforms at the probed pg_pins are saved in the FSDB file during dynamic analysis.

EXAMPLES

The following example shows how to set probes at the specified list of pg_pins.

```
pr_shell> set_probes [get_pg_pins INV*/vdd]
Information: Voltage and current probes are installed at 12 pg_pins.
1
```

The following example shows how to set probes at the pg_pins of the supply net, VDD.

```
pr_shell> set_probes [get_pg_pins -of_object VDD]
Information: Voltage and current probes are installed at 128 pg_pins.
1
```

SEE ALSO

```
report_probes(2)
remove_probes(2)
calculate_rail_voltage(2)
```

set_program_options

Sets options that affect the performance and capacity.

SYNTAX

```
status set_program_options
   [-enable_high_capacity]
   [-disable_high_capacity]
   [-enable_fast_analysis]
```

ARGUMENTS

-enable_high_capacity

Runs the tool in high capacity mode, which reduces the peak memory footprint.

-disable_high_capacity

Disables the high capacity mode of the program and returns to the default normal analysis mode, which usually increases the peak memory footprint.

-enable_fast_analysis

Enables the fast analysis mode of the program. Performance and capacity might be improved. By default, fast analysis mode is disabled.

DESCRIPTION

This command provides options that trade off performance and capacity.

High Capacity Mode

In high capacity mode, the tool temporarily stores data to a local disk partition specified by the **pt_tmp_dir** variable; the default is the /tmp directory. When you exit the session, the consumed disk space is automatically released. For effective use of this mode, the capacity of the local disk on the host machine should be least two times larger than the physical RAM. The potential capacity improvement might not be fully realizable in case of insufficient disk space.

Using high capacity mode does not change the timing analysis results. You can enable or disable high capacity mode only when there are no designs and libraries loaded in the program memory.

By default, high capacity mode is enabled. To see whether high capacity mode is enabled, check the setting of the **sh_high_capacity_enabled** read-only variable.

To specify the actual capacity improvement in high capacity mode, set the **sh_high_capacity_effort** variable.

Fast Analysis Mode

Fast analysis mode provides high performance and accuracy for early stage timing. It overrides some variables and options to configure the tool for higher performance during early analysis runs. In fast analysis mode, path-based analysis is turned off.

By default, fast analysis mode is disabled. To see whether fast analysis mode is enabled, check the setting of the **sh_fast_analysis_mode_enabled** read-only variable.

EXAMPLES

The following command enables high capacity mode with the default settings:

```
pr_shell> set sh_high_capacity_effort high
Information: The setting will be effective after next 'set_program_options' command. (PTHC-001)
high

pr_shell> set_program_options -enable_high_capacity
Information: enabling high capacity analysis in 'high' effort........ (PTHC-001)
pr_shell> printvar sh_high_capacity_enabled
sh_high_capacity_enabled = "true"
```

The following command enables fast analysis mode:

```
pr_shell> set_program_options -enable_fast_analysis
Information: Fast analysis mode is enabled. (PTANA-002)

pr_shell> printvar sh_fast_analysis_mode_enabled
sh_fast_analysis_mode_enabled = "true"
```

set_propagated_clock

Specifies propagated clock latency.

SYNTAX

```
string set_propagated_clock
   object_list
```

Data Types

object_list	list

ARGUMENTS

object_list

Specifies a list of clocks, ports, or pins.

DESCRIPTION

Specifies that delays be propagated through the clock network to determine latency at register clock pins. If it is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a specified latency (designated by the **set_clock_latency** command), or zero latency, by default. Propagated clock latency is used for post-layout, after final clock tree generation. Ideal clock latency provides an estimate of the clock tree for pre-layout.

If the **set_propagated_clock** command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. If it is applied to an object (clock, port, or pin) on which network latency is already defined, then a warning is issued.

Virtual clocks do not have any source, and they cannot affect any register in the design. Thus, virtual clocks cannot be made propagated, and an error is issued if the **set_propagated_clock** command is applied on a virtual clock.

To undo the **set_propagated_clock** command, use either the **remove_propagated_clock** or **set_clock_latency** command to provide an ideal latency.

To see propagated clock attributes on clocks, use the **report_clock** command.

Limitations: CRPR does not support propagated clocks set on pins or ports unless they are source pins or ports of common clock for a path. If the **timing_remove_clock_reconvergence_pessimism** variable is set to TRUE, then propagated clocks should be defined on clock objects, not pins or ports.

EXAMPLES

The following example specifies using propagated clock latency for all clocks in the design.

```
pt_shell> set_propagated_clock [all_clocks]
```

SEE ALSO

```
timing_remove_clock_reconvergence_pessimism(3)
remove_propagated_clock(2)
report_clock(2)
set_clock_latency(2)
```

set_pulse_clock_max_transition

Sets the maximum transition for pulse generator input and pulse clock network with respect to the main library trip-points.

SYNTAX

```
string set_pulse_clock_max_transition
   transition_value
   [-rise]
   [-fall]
   [-transitive_fanout]
   object_list
```

Data Types

transition_value	float
object_list	list

ARGUMENTS

-rise

Specifies rising transition for all selected pins.

-fall

Specifies falling transition for all selected pins.

-transitive_fanout

Specifies the constraint is set at the transitive fanout of pulse generator. If this option is not set, only the input of the pulse generators are constrained.

transition_value

Sets the transition limit (Value >= 0). This is the maximum transition time in main library time units.

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design. .

DESCRIPTION

Specifies the maximum transition in the pulse clock network at the forward of pulse generator cell, pulse

generator lib cell, clock and design if *-transitive_fanout* option is set. By default, the maximum transition constraint is set at the input of the pulse generator of the specified object list. If the cell or lib cell is not a pulse generator, or if the clock or design does not have any pulse generators, then this maximum transition limit has no effect. The pulse clock network or the input of pulse generator is expected to have transition time less than the specified the *transition_value* option.

The maximum transition constraint set on design by the **set_max_transition** command applies to the pulse clock network and input of pulse generator as also maximum transition constraint set on the pins in the library. Transition constraints are interpreted in main library trip-points and derate. During slack computation, the constraints are scaled to the pin trip-points and derate. In the case of conflicting constraints on a pin, the tightest constraint applies.

The maximum transition constraint can be optionally restricted to rising transitions only or falling transitions only by using the *-rise* or *-fall* options respectively. If the *-rise* or *-fall* options are not specified, both rise and falling transitions are constrained.

The **report_constraint -pulse_clock_max_transition** and **report_pulse_clock_max_transition** commands show the maximum transition constraint evaluations. The limit with appropriate scaling used for slack computation is reported.

To remove maximum pulse clock transition limits, use the **remove_pulse_clock_max_transition** command.

EXAMPLES

The following example sets a maximum transition limit of 0.4 units on input of all the cells corresponding to lib cell pulse_rise_high in library celllib.

```
pt_shell> set_pulse_clock_max_transition 0.4 {"celllib/pulse_rise_high"}
```

The following example sets a maximum transition limit of 0.4 units on all the pulse clock networks in the clock network clk.

```
pt_shell> set_pulse_clock_max_transition -transitive_fanout 0.4 [get_clocks clk]
```

SEE ALSO

```
current_design(2)
remove_pulse_clock_max_transition(2)
report_pulse_clock_max_transition(2)
report_constraint(2)
```

set_pulse_clock_max_width

Sets the maximum pulse width constraint for pulse generator network.

SYNTAX

```
string set_pulse_clock_max_width
   [-transitive_fanout]
   pulse_width_value
   object_list
```

Data Types

```
pulse_width_value      float
object_list            list
```

ARGUMENTS

-transitive_fanout

The constraints are applied to the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

pulse_width_value

Sets the pulse width limit (Value >= 0).

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

DESCRIPTION

Specifies the maximum pulse width limit for the pulse clock network at the forward of pulse generator cell by pulse generator instance name or pulse generator library cell. Setting the pulse width constraint by clock applies the constraint to the forward of all pulse generators driven by the clock. Setting by pulse width constraint by design applies the constraint to all the pulse clock networks in the design. In the presence of a conflicting constraint, the most restrictive constraint is used. If the cell or lib cell is not a pulse generator, or clock or design does not have any pulse generators, then this maximum width limit has no effect. The pulse clock network is expected to have pulse width time less than the specified *pulse_width_value* option.

The high or low pulse width constraint is inferred based on sense propagation. For example, after an inverter, a high pulse width constraint is converted to a low pulse width constraint.

The **report_constraint -pulse_clock_max_width** and **report_pulse_clock_max_width** commands show maximum pulse width constraint evaluations.

To remove maximum pulse width limits, use the **remove_pulse_clock_max_width** command.

EXAMPLES

The following example sets a maximum pulse width limit of 0.4 units on all the pulse clock networks in the design.

```
pt_shell> set_pulse_clock_max_width 0.4 [current_design]
```

SEE ALSO

```
current_design(2)
remove_pulse_clock_max_width(2)
report_pulse_clock_max_width(2)
report_constraint(2)
```

set_pulse_clock_min_transition

Sets the minimum transition at the input of pulse generator with respect to the main library trip-points.

SYNTAX

```
string set_pulse_clock_min_transition
   transition_value
   [-rise]
   [-fall]
   object_list
```

Data Types

```
transition_value      float
object_list           list
```

ARGUMENTS

-rise

Specifies rising transition for all selected pins.

-fall

Specifies falling transition for all selected pins.

transition_value

Sets the transition limit (Value >= 0). This is the minimum transition time in main library time units.

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

DESCRIPTION

Specifies the minimum transition at the input of pulse generator cell and pulse generator lib cell. If the constraint is specified on the clock and design, all the inputs of pulse generators in the clock network and design respectively are constrained. If the cell or lib cell is not a pulse generator or clock or design does not have any pulse generators, then minimum pulse clock transition limit has no effect. The input of pulse generator is expected to have transition time greater than the specified *transition_value* option.

Transition constraints are interpreted in main library trip-points and derate. During slack computation, the

constraints are scaled to the pin trip-points and derate. In the case of conflicting constraints on a pin, the tightest constraint applies.

The minimum transition constraint can be optionally restricted to rising transitions only or falling transitions only by using the *-rise* or *-fall* options, respectively. If the *-rise* or *-fall* options are not specified, both rise and falling transitions are constrained.

The **report_constraint -pulse_clock_min_transition** and **report_pulse_clock_min_transition** commands show minimum transition constraint evaluations. The constraint with appropriate scaling used for slack computation is reported.

To remove minimum pulse clock transition limits, use the **remove_pulse_clock_min_transition** command.

EXAMPLES

The following example sets a minimum transition limit of 0.4 units on input of all the cells corresponding to lib cell pulse_rise_high in library celllib.

```
pt_shell> set_pulse_clock_min_transition 0.4 {"celllib/pulse_rise_high"}
```

SEE ALSO

current_design(2)
remove_pulse_clock_min_transition(2)
report_pulse_clock_min_transition(2)
report_constraint(2)

set_pulse_clock_min_width

Sets minimum pulse width constraint for pulse generator network.

SYNTAX

```
string set_pulse_clock_min_width
   pulse_width_value
   [-transitive_fanout]
   object_list
```

Data Types

pulse_width_value	float
object_list	list

ARGUMENTS

-transitive_fanout

The constraints are applied to the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

pulse_width_value

Sets the pulse width limit (Value >= 0).

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

DESCRIPTION

Specifies the minimum pulse width limit for the pulse clock network at the forward of pulse generator cell by pulse generator instance name or pulse generator library cell. Setting the pulse width constraint by clock applies the constraint to the forward of all pulse generators driven by the clock. Setting by pulse width constraint by design applies the constraint to all the pulse clock networks in the design. Minimum pulse width constraint from the library also applies to the pulse clock network. In the presence of a conflicting constraint, the most restrictive constraint is used. If the cell or lib cell is not a pulse generator or clock or design does not have any pulse generators, then this minimum width limit has no effect. The pulse clock network at the forward of pulse generator is expected to have width more than the specified *pulse_width_value* option.

The high or low pulse width constraint is inferred based on sense propagation. For example, after an inverter, a high pulse width constraint is a low pulse width constraint.

The **report_constraint -pulse_clock_min_width** and **report_pulse_clock_min_width** commands show minimum pulse width constraint evaluations.

To remove minimum pulse width limits, use the **remove_pulse_clock_min_width** command.

EXAMPLES

The following example sets a minimum pulse width limit of 0.4 units on all the pulse clock networks in the clock network clk:

```
pt_shell> set_pulse_clock_min_width 0.4 [get_clocks clk]
```

SEE ALSO

```
current_design(2)
remove_pulse_clock_min_width(2)
report_pulse_clock_min_width(2)
report_constraint(2)
```

set_qtm_attribute

Sets an attribute to the specified value on QTM object(s).

SYNTAX

```
string set_qtm_attribute
   -class class_name
   attr_name
   value
   [object_names]
```

Data Types

```
class_name        string
attr_name         string
value             string
object_names      list
```

ARGUMENTS

-class *class_name*

Specifies the class of the QTM objects to set attribute on. Allowed values are **lib**, **lib_cell**, or **lib_pin**.

attr_name

Specifies the name of the attribute.

value

Specifies the value of the attribute.

object_names

Specifies the names of the objects on which to set the attribute. Each element in the list is a name to identify an object in the specified class. The object names should not be specified when the object class is either a **lib** or **lib_cell** value. It must be specified when the class is a **lib_pin** value, in which case the object names should be the names of ports already defined for the QTM.

DESCRIPTION

The **set_qtm_attribute** command sets attributes on objects. These attributes are either application attributes that are reserved by PrimeTime or defined with the **define_qtm_attribute** command as the user attributes.

Please note the order of the command arguments. Also note that PrimeTime the **list_attribute**, **report_attribute**, and **get_attribute** commands do not work on attributes defined or set on the QTM objects.

EXAMPLES

The following example defines a Boolean attribute 'is_XYZ' for QTM cell, then sets the value on the QTM cell under construction.

```
pt_shell> define_qtm_attribute -type boolean -class lib_cell "is_XYZ"
pt_shell> set_qtm_attribute -class lib_cell "is_XYZ" "true"
```

The following example sets the attribute 'max_transition' for 2 QTM ports. Because 'max_transition' is an application reserved attribute for the object class 'lib_pin', it should be set directly.

```
pt_shell> set_qtm_attribute -class lib_pin "max_transition" 1.0 {IN, CLK}
```

SEE ALSO

define_qtm_attribute(2)
remove_qtm_attribute(2)
define_user_attribute(2)

set_qtm_global_parameter

Sets a global parameter for a quick timing model.

SYNTAX

```
string set_qtm_global_parameter
   [-param setup | hold | recovery |removal | clk_to_output]
   [-lib_cell lib_cell]
   [-pin pin_name]
   [-clock pin_name]
   [-value parameter_value]
```

Data Types

```
lib_cell            string
pin_name            string
parameter_value     float
```

ARGUMENTS

-param setup | hold | recovery |removal | clk_to_output

Sets one of the following global parameters: **setup**, **hold**, **recovery**, **removal** or **clk_to_output**.

-lib_cell *lib_cell*

Specifies the library cell that you want to use to mimic the global parameter.

-pin *pin_name*

Specifies the related pin for the global parameter if you are using the **-lib_cell** option to mimic the behavior. If the parameter is **setup** or **hold**, this pin must be an input pin. If the parameter is **clk_to_output**, this pin must be an output pin. If you do not use this option, the quick timing model uses the first encountered input pin that is constrained, in the case of the **setup** or **hold** parameter; and the first output pin that has an edge delay arc coming into it, in the case of the **clk_to_output** parameter.

-clock *pin_name*

Specifies the constraining pin in the case of **setup** or **hold**; and the launch pin in the case of **clk_to_output** parameter.

-value *parameter_value*

The value of the global parameter in time units.

DESCRIPTION

This command sets the value of a quick timing model (QTM) global parameter. The global parameter can be either **setup**, **hold**, , **recovery**, **removal** or **clk_to_output**. These parameters specify the global parameter which is added to the arcs. To the QTM setup/hold/recovery/removal arc, the time is added to arrive at the final delay. To the QTM edge triggered delay arc the global clk_to_output time is added to arrive at the final delay.

You can specify the global parameters of the sequential element by specifying a cell in the library (the **-lib_cell** option), or by giving the actual value, by using the **-value** option. If the **-lib_cell** option is specified, you can specify the clock pin and the input pin name. If you do not specify one, the first setup, hold, or clk_to_output arc encountered is chosen (based on which parameter is being set). If only the **-clock** option is specified, the first setup, hold, or clk_to_output arc (depending on the global parameter) from that clock pin is chosen. If only the **-pin** option is specified, the first setup, hold, or clk_to_output arc (depending on the type of global parameter) coming into the pin is chosen.

To report information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

In the following examples, before using the **-lib_cell** option, you must load the technology library and set the QTM technology. To do this, use the following sequence of commands:

```
read_db my_technology_library.db
set_qtm_technology -library my_technology_library
```

The following example sets the global setup time equivalent to the setup time of library cell DFF1:

```
pt_shell> set_qtm_global_parameter -param setup -lib_cell DFF1
```

The following example sets the global hold time equivalent to the setup time of pin D of library cell DFF1:

```
pt_shell> set_qtm_global_parameter -param setup -lib_cell DFF1 -pin D
```

The following example sets the global clk_to_out time to be 1.5 time units:

```
pt_shell> set_qtm_global_parameter -param clk_to_output -value 1.5
```

SEE ALSO

```
create_qtm_constraint_arc(2)
create_qtm_delay_arc(2)
create_qtm_drive_type(2)
create_qtm_load_type(2)
create_qtm_model(2)
create_qtm_path_type(2)
report_qtm_model(2)
save_qtm_model(2)
```

set_qtm_port_drive

Sets the drive on the quick timing model port.

SYNTAX

```
string set_qtm_port_drive
   [-type drive_type]
   [-value drive_value]
   [-input_transition_rise rtrans]
   [-input_transition_fall ftrans]
   [-subtract_max_delay_from_total]
   port_list
```

Data Types

drive_type	string
drive_value	float
rtrans	float
ftrans	float
port_list	list

ARGUMENTS

-type *drive_type*

Specifies the global *drive_type* parameter.

-value *drive_value*

Specifies the drive value in terms of drive resistance units.

port_list

Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of quick timing model ports or a pattern that matches quick timing model ports.

-input_transition_rise *rtrans*

Specifies the input rising transition time associated with the **-input_pin** option of the specified drive type to compute the delay for the drive arc for this port. If you do not use this option, the delay table for rising input of the drive arc is loaded from library as is.

Use the **-input_transition_rise** and **-input_transition_fall** options to capture the transition time associated with the **-input_pin** option defined for the drive type referred. This can obtain more accurate information on the transition time and delay time for the drive arc defined for this port.

-input_transition_fall *ftrans*

Specifies the input falling transition time associated with the **-input_pin** option of the specified drive type to compute the delay for the drive arc for this port. If you do not use this option, the delay table for falling input of the drive arc is loaded from library as is.

-subtract_max_delay_from_total

Indicates that the drive arc delay computed with max load is subtracted from the total delay budgeted to this output port. This affects all types of delay arcs that reach this output port. By default, the drive arc delay is added on top of the quick timing model delay arcs created to the output port and therefore increase the actual delay.

DESCRIPTION

This command sets the drive of the output port of the quick timing model. There are two methods for setting the drive:

- Use one of the global *drive_type* parameters, if you have drive types defined.
- Define the drive in terms of the drive resistance units.

When drive is defined for an output port, by default, the actual delay to the output port is the delay arc defined by the **create_qtm_delay_arc** command for the port plus the delay computed for this drive arc. If in your QTM creation, the delay defined by the **create_qtm_delay_arc** command for the port is actually the maximum delay budgeted for the output and you do not want the drive arc to add extra delay on top of that budget, you can optionally use the **-subtract_max_delay_from_total** option to keep the delay with maximum load at port unchanged. But with smaller load at the output port, the delay might actually be smaller than the budgeted maximum delay.

To define drive types, use the **create_qtm_drive_type** command.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command sets the drive of the OUT1 output port to be the drive1 drive type.

```
pt_shell> set_qtm_port_drive -type drive 1 OUT1
```

The following command sets the drive of the OUT2 output port to be 5 drive units.

```
pt_shell> set_qtm_port_drive -value 5.0 OUT2
```

SEE ALSO

```
create_qtm_load_type(2)
create_qtm_model(2)
report_qtm_model(2)
```

```
save_qtm_model(2)
get_qtm_ports(2)
set_qtm_port_load(2)
```

set_qtm_port_load

Sets the load on quick timing model (QTM) ports.

SYNTAX

```
string set_qtm_port_load
   [-type load_type]
   [-factor multiplication_factor]
   [-value load_value]
   port_list
```

Data Types

load_type	string
multiplication_factor	float
load_value	float
port_list	list

ARGUMENTS

-type *load_type*

Specifies the global *load_type* parameter.

-factor *multiplication_factor*

Specifies the multiplication factor for the load type to set the load on the quick timing model port.

-value *load_value*

Specifies the load value in the capacitance units.

port_list

Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of quick timing model ports or a pattern that matches quick timing model ports.

DESCRIPTION

This command sets the load of the input port of the quick timing model. There are two methods for setting the load:

- Use the global *load_type* parameter along with a multiplication factor if you have load types defined.

- Define the load in terms of the capacitance units.

To define load types, use the **create_qtm_load_type** command.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following command sets the load on IN1 to be 2 times the global *load_type* load1.

```
pt_shell> set_qtm_port_load -type load1 -factor 2 IN1
```

The following command sets the load on IN2 to be 4.5 capacitance units.

```
pt_shell> set_qtm_port_load -value 4.5 IN2
```

SEE ALSO

```
create_qtm_load_type(2)
create_qtm_model(2)
get_qtm_ports(2)
report_qtm_model(2)
save_qtm_model(2)
set_qtm_port_drive(2)
```

set_qtm_technology

Sets the quick timing model (QTM) technology variables.

SYNTAX

```
string set_qtm_technology
   [-library name]
   [-max_transition trans_value]
   [-min_transition trans_value]
   [-max_capacitance cap_value]
   [-min_capacitance cap_value]
   [-wire_load_model wlm_name]
   [-operating_condition opcond_name]
   [-process process_value]
   [-voltage voltage_value]
   [-temperature temperature_value]
```

Data Types

```
name            string
trans_value     float
cap_value       float
wlm_name        string
opcond_name     string
process_value   float
voltage_value   float
temperature_value   float
```

ARGUMENTS

-library *name*

Specifies the library name for using library elements to define global parameters.

-max_transition *trans_value*

Specifies the maximum transition value.

-min_transition *trans_value*

Specifies the minimum transition value.

-max_capacitance *cap_value*

Specifies the maximum capacitance value.

-min_capacitance *cap_value*

Specifies the minimum capacitance value.

-wire_load_model ***wlm_name***

Specifies the wire load model used while calculating delays.

-operating_condition ***opcond_name***

Specifies the default operating condition name for the quick timing model in a multicorner, multimode (MCMM) design. Must be specified together with **-process**, **-voltage**, and **-temperature**.

-process ***process_value***

Specifies the process scaling factor for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**. Allowed values are 0.0 through 100.0.

-voltage ***voltage_value***

Specifies the voltage value, in Volts, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

-temperature ***temperature_value***

Specifies the temperature value, in degrees Celsius, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: **-operating_condition**, **-process**, **-voltage**, and **-temperature**.

DESCRIPTION

This command sets the various quick timing model technology parameters. Use the **-library** option if you use library cells to define path_types, drive_types, load_types, or if you set the other global parameters in terms of library cells. Before setting the library, you must ensure that the library is loaded.

If you use the **-wire_load_model** option, the path_type delays are calculated using the information. You can use the **-operating_condition** option to specify the default operating condition name for the quick timing model in a multicorner, multimode (MCMM) design. It must be specified together with **-process**, **-voltage**, and **-temperature** options.

To show the information about the current quick timing model, use the **report_qtm_model** command.

For basic information about quick timing models, see the **create_qtm_model** man page.

EXAMPLES

The following example sets the library to my_lib, which is in the my_lib.db file.

```
pt_shell> read_db my_lib.db
pt_shell> set_qtm_technology -library my_lib
```

The following command sets the **-wire_load_model** option to wlm1.

```
pt_shell> set_qtm_technology -wire_load_model wlm1
```

SEE ALSO

```
create_qtm_model(2)
report_qtm_model(2)
save_qtm_model(2)
```

set_query_rules

Defines rules for rule-based query in golden UPF mode. In non-golden UPF mode, this command is ignored.

SYNTAX

```
status set_query_rules
   [-hierarchical_separators separator_list]
   [-bus_name_notations bus_name_list]
   [-class class_list]
   [-regsub regsub_list]
   [-regsub_cumulative]
   [-wildcard]
   [-suffix suffix_list]
   [-verbose]
   [-nocase]
   [-reset]
```

Data Types

```
separator_list       list
bus_name_list        list
class_list           list
regsub_list          list
suffix_list          list
```

ARGUMENTS

-hierarchical_separators *separator_list*

Specifies a list of equivalent hierarchy separators in object names. There must be at least two elements in the list. The default list is {/ _ .}. The elements are preferably all single character, although at most one multicharacter element is allowed. No more than one alphabetical character, a-z or A-Z, is allowed as a hierarchical separator. The single alphabetical character element is also mutually exclusive with the multicharacter element. The following special characters are not supported as hierarchy separators nor they are allowed as one of the characters for the multicharacter hierarchy separator: 0-9, *, ?, \, +, ^, [,], (,), <, >, and {, }.

-bus_name_notations *bus_name_list*

Specifies a list of equivalent bus name notations in object names. There must be at least two elements in the list, and the list element must be of two characters exactly. The first character of the element is the opening bus name character, and the second character is the closing bus name character. The default list is {[] __ ()}. No more than one matching pair of alphabetical characters a-z or A-Z is allowed as bus name notations. The following characters are not supported as opening or closing bus name characters: 0-9, *, ?, \, +, ^, and /. Bracketed bus notations must be paired. For example, "[)" are not properly paired brackets, therefore it is not supported. For non-bracket bus name notations, the opening and closing bus name character must be the same. For example, "-_" is not a supported bus name notation.

-class *class_list*

Specifies a list of object class names for which the query rules are applied. Object classes cell, port, pin, net, power_domain, supply_net, supply_set, supply_port and power_switch are supported. The default is to enable all of them, however you could specify a subset of them.

-regsub *regsub_list*

Specifies a list of options for the **regsub** Tcl command. Each list should include three string elements: {{*options*} {*match_regexp*} {*substitution_exp*}}. Query uses the **-regsub** option to evaluate the object name and pattern strings during the matching process.

The **-regsub** option can be specified multiple times in one **set_query_rules** command. When multiple **-regsub** options are used, the **-regsub** options are applied to the leaf object name and pattern in the order you specify.

The **-regsub** options applies only to pin, port, net, and leaf cells. If **-regsub** is used without other query rules, the default values for the other options are still used. To save runtime, use this option only if the existing rule-based matching scheme fails to match an object.

In addition to the Tcl built-in options for the command **regsub**, the following options are supported:

- **-class** specifies a single object class name such as cell, port pin or net. The **-class** option makes the **-regsub** query rule object specific. Object class specific **-regsub** rule is the preferred to legacy global **-regsub** rule. The latter has been deprecated. For backward compatibility purposes, the global **-regsub** query rule is automatically converted to the class specific rules for port, pin and net, but not for cell class.
- **-cumulative** specifies that the multiple **-regsub** rules for the same object class are executed with cumulative effect in the sense that the subsequent **-regsub** rule is applied to the outputs of the object name and pattern string from the previous **-regsub** rules. After this option is used for an object class in one **-regsub**, all **-regsub** rules for that object class have the cumulative option.

-regsub_cumulative

Although this legacy option is supported, you should use **-regsub -cumulative** instead. If you use the **-regsub_cumulative** option, the tool automatically converts to the class-specific **-cumulative** option for port, pin, and net, but not for cell class.

-wildcard

Enables wildcard support in rule-based match. Be cautious with this option because this option can end up matching more objects after ungroup and change_names.

-suffix *suffix_list*

Enables suffix name rule. Only the predefined suffix *_reg* and *_tri* are supported. Suffix *_reg* is only applicable to register cells. When enabled, a cell name pattern such as U4 would match the U4_reg register cell. Suffix *_tri* is only applicable to tristate cells. When enabled, a cell name pattern such as U8 would match the U8_tri tristate cell object.

-verbose

Prints messages indicating the object is matched using query rules.

-nocase

Enables case-insensitive rule-based match. This is the preferred way of enabling case-insensitive rule-based match.

-reset

Resets query rules to the program default.

DESCRIPTION

The **set_query_rules** defines query rules to guide the object queries in golden UPF mode. The purpose of the rule-based matching is to resolve the naming differences caused by altering names in Design Compiler and IC Compiler. The hierarchy separators and bus names are typical sources of the naming differences.

When a list of characters is defined as equivalent in terms of hierarchy separator, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {/ _ .} is defined as equivalent hierarchy separators, the cell named "a.b_c/d_e" is matched with name string "a/b_c.d/e" provided in the golden UPF files.

When a list of bus notations is defined as equivalent in terms of bus names, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {[] __} is defined as equivalent bus notations, the cell named "a[4][5]" is matched with the name string "a_4__5_" provided in the golden UPF files.

When the **-regsub** rules are specified, Tcl **regsub** command applies the rules to both the object name and the pattern string for rule-based matching if the other rule-based matching scheme fails to match an object.

Be very cautious about enabling wildcard support in rule-based match, because the greedy nature of the wildcard match could often give you more matches than you want.

Use the **-reset** option to reset the query rules to the program default. Use **-show** to report the current query rules. When **-show** is used with other options, the new rules are set first and then reported.

The **-nocase** option of command **set_query_rules** is the preferred way of enabling case-insensitive rule-based matching, although global variable **find_ignore_case** variable is honored. For runtime reasons, avoid using the **-nocase** option in query commands such as **get_cells** or **get_pins**.

The **set_query_rules** command can be issued multiple times. Unless the **-show** option is used alone, the new rules always overwrite the previous rule settings.

Be cautious when defining query rules. To save runtime, do not use unnecessary query rules.

The **set_query_rules** command is effective only in golden UPF mode. Runtime for rule-based matching could be much slower.

EXAMPLES

```
# Show program default
prompt> set_query_rules -show

*************************
Query Rules
*************************

          Rule Type  Rule Value
-------------------  --------------------
Hierarchy-Separator  {/ _ .}
       Bus-Notation  {[] __ ()}
       Object-Class  {cell port pin net power_domain power_switch supply_net supply_set supply_port}
             Nocase  false
```

```
            Wildcard  false
             Verbose  false
1


# Non-default
prompt> set_query_rules \
    -hierarchical_separators {/ _} \
    -bus_name_notations {[] ||} \
    -class {cell port} -show

*************************
Query Rules
*************************

            Rule Type   Rule Value
  -------------------   --------------------
  Hierarchy-Separator   {/ _}
         Bus-Notation   {[] ||}
         Object-Class   {cell port}
               Nocase   false
             Wildcard   false
              Verbose   false
1

# Use alphabetical characters
prompt> set_query_rules \
    -hierarchical_separators {/ x} \
    -class {cell port} \
    -bus_name_notations {[] __ || xx} -show

*************************
Query Rules
*************************

            Rule Type   Rule Value
  -------------------   --------------------
  Hierarchy-Separator   {/ x}
         Bus-Notation   {[] __ || xx}
         Object-Class   {cell port}
               Nocase   false
             Wildcard   false
              Verbose   false
1

# Unsupported usage
prompt> set_query_rules \
    -hierarchical_separators {/ x y} \
    -bus_name_notations {[] __ || xx}
Error: At most 1 alphabetical or multicharacter hierarchy separator is allowed. (UID-1067)
0
```

SEE ALSO

load_upf(2)

set_rail_current

Assigns electrical current to the PG pins of cells to be used in rail analysis.

SYNTAX

```
status set_rail_current
   object_list
   [-static current_value]
   [-waveform current_waveform]
   -lib_cell library_cell
   [-pg_pin pg_pin]
   [-leakage leakage_current]
   [-when_condition when_condition]
   [-input_switching_condition input_pin_switching_condition]
   [-fsdb_file fsdb_file]
   [-fsdb_signal_name fsdb_pin]
```

Data Types

```
list object_list
float current_value
list current_waveform
list library_cell
string pg_pin
float leakage_current
string when_condition
list input_switching_condition
string fsdb_file
string fsdb_signal_name
```

ARGUMENTS

object_list

Specifies a list of PG pins to which electrical current is assigned. Each item in the list can be a PG name pattern or collection.

-static *current_value*

Specifies the static electrical current value to be assigned, in unit of the technology library (which is the first .db defined in link_path). Use this option with the *object_list* option.

This option can only be used after the **open_mw_cel** or **open_block** command.

-waveform *current_waveform*

Specifies the dynamic current waveform to be assigned. When the command is invoked before the design is linked, by default the time and current units are second and Amp, respectively. When the command is invoked

after the design is linked, all units are in technology library units.

-lib_cell ***library_cell***

Specifies a list library master cells on which PG pins will be set with current.

This option has to be used with the *-pg_pin* option, and can only be used before the **open_mw_cel** and **open_block** command.

This option is mutually exclusive with the *object_list* option.

-pg_pin ***pg_pin***

Specifies a PG pin name on which rail current is set. This option has to be used with the *-lib_cell* opiton.

-leakage ***leakage_current***

Specifies leakage current for a library cell specified in the *-lib_cell* option. The default current unit is Amp. This option can only be used before the **open_mw_cel** command.

This option cannot be used with the *-input_switching_condition*, or *-when_condition* options.

-when_condition ***when_condition***

Specifies a when condition for the current or current waveform set in the command. This option, if specified, has to be used together with the options *-lib_cell* and *-pg_pin*.

-input_switching_condition ***input_switching_condition***

Specifies input pin switching condition for the current or current waveform set in the command. The pin condition has to follow the format "pin pin_condition". pin_condition can be either "r" (to represent rising condition) or "f" (to represent falling condition). This option, if specified, has to be used together with the options *-lib_cell* and *-pg_pin*.

-fsdb_file ***fsdb_file***

Specifies FSDB file from which a current waveform can be extracted. This option can be used with the option *-lib_cell* or *object_list*. When this option is used, the current and time units are the ones used in FSDB file, which are usually in second and Amp.

-fsdb_signal_name ***fsdb_signal_name***

Specified FSDB signal name from which current waveform can be extracted. This option has to used with the option *-fsdb_file*.

DESCRIPTION

This command assigns electrical current to a power and ground (PG) pin. The command returns 1 if successful, and 0 if unsuccessful. The command can be used either before or after the design is linked. When the command is used before the design is linked, the default units for time and current are second and Amp. Otherwise the technology library unit is used.

When the command is used before the design is linked, users can set current waveforms, or leakage currents for a list of library cells using the *-lib_cell* option.

The PG pin current is typically calculated as a result of the **update_currents** command run, which derives the electrical current for a PG pin from the actual events on the cell for the VCD-based flow or from the statistical

event probabilities, based on power models for the cell. (The power models are NLPM or CCSP for standard cells and macro models for other non-standard logic library cells.) When there is no power model for a cell, no currents can be generated. This command provides a mechanism for assigning currents when power models are absent, or to override the calculated current values.

Current can be assigned as an aggregated time-average static current value with the *-static* option, or current waveform in terms of a list of time and current values {t1 i1 t2 i2 ...} with the *-waveform* option. The command overwrites any current previously assigned by the user.

When static current is assigned to the PG pin, the corresponding power value for this PG pin is also assigned. This is equivalent of executing the **set_rail_power** command where the power value is calculated as static current multiplied by PG pin voltage. This makes the power value consistent with static current for the given PG pin. Thus, users can use the **report_rail_power** command to query the assigned power value.

When static current is assigned to the power pin, static current for the ground pin will be derived from the current assigned to the power pin by using the library power-to-ground pin mapping. This ensures that the currents on the power pins of a cell are consistent with respect to each other, such that the total static current on a cell's power and ground pins is the arithmetic inverse of the total static current on the corresponding ground pins.

The static current assigned by this command and its related power value always override the values calculated during the **update_currents** command, irrespective of their execution order. The **set_rail_current** command has higher precedence than the **update_currents** command.

Note that the **set_rail_current** command with the *-static* option has the same effect and precedence as the **set_rail_power** command, when they apply to the same PG pin. Thus, the **report_rail_power** command can be used to report the power assigned by both commands.

Users can determine the electrical currents of the PG pin by querying the PG pin *static_current* or *current_waveform* attribute. The current assigned by the **set_rail_current** command can be removed by the **remove_rail_current** command.

EXAMPLES

The following example specifies a PWL current waveform for an INV64 library cell on PG pin VDD. The current unit is in Amp, and time unit is in second.

```
pr_shell> set_rail_current -waveform {0.0 0.0 0.1e-12 0.1e-3 0.2e-12 0.0} -lib_cell INV64 -pg_pin VDI
pr_shell> open_mw_cel -lib my_lib my_design
```

The following example specifies a PWL current waveform extracted from a FSDB file for library cell INV64 on PG pin VDD. The FSDB signal name is instance1/VDD/current. Note the current and time read from the FSDB file are in units of second and Amp.

```
pr_shell> set_rail_current -fsdb_file instance.fsdb -fsdb_signal_name instance1/VDD/current -lib_cell
pr_shell> open_mw_cel -lib my_lib my_design
```

The following example specifies a leakage current 0.0001mA for a library DECAP64 on PG pin VDD.

```
pr_shell> set_rail_current -leakage 0.1e-6 -lib_cell INV64 -pg_pin VDD
pr_shell> open_mw_cel -lib my_lib my_design
```

The following example applies an electrical current of 5 milliamps to a PG pin after the design is linked.

```
pr_shell> open_mw_cel -lib my_lib my_design
...
```

```
pr_shell> report_units
...
Current_unit           : 0.001 Amp
...
pr_shell> set_rail_current -static -5 [get_pg_pin MyCell/VDDL]
1
pr_shell> get_attribute [get_pg_pin MyCell/VDDL] static_current
-5
```

The following example assigns current waveform to the vdd pin of instance U153 after design is linked.

```
pr_shell> open_mw_cel -lib my_lib my_design
...
pr_shell> set_rail_current -waveform {0.0 0.0 1.0 -1.0 2.0 0.0} U153/vdd
```

SEE ALSO

rail_attributes(2)
report_rail_current(2)
remove_rail_current(2)
set_rail_power(2)
update_currents(2)

set_rail_macro_mode_sequence

Assigns a macro mode sequence to the PG pins of hard macros for vectorfree and vector-based dynamic rail analysis.

SYNTAX

```
status set_rail_macro_mode_sequence
   [-vectorfree_modes t1 m1 t2 m2...]
   [-clock associated_clock_name]
   [-import import_file_name]
   [time_mode_list]
   [object_list]
```

Data Types

```
string \flassociated_clock_name
string \flimport_file_name
list time_mode_list
list mode_only_list
list object_list
```

ARGUMENTS

-vectorfree_modes

Specifies a list of mode names to be applied to the specified PG pins during vector free rail analysis. The time_mode_list and -vectorfree_modes options are mutually exclusive.

-clock

Specifies the clock to trigger the specified macro model. When not specified, the tool uses the clock event for triggering the modes in the vectorfree mode.

-import

Specifies the name of the file that contains the model sequence data. Note that the settings specified by other options always have the higher priority than those in the model sequence file.

Here is an example of the model sequence file:

M1/V* READ STANDBY WRITE READ READ M2/V* WRITE WRITE READ STANDBY WRITE

time mode list

Specifies a list of mode names to be applied to the specified macro model in the following syntax:

{t1 m1 t2 m2 ...}

Note that at least one mode is needed. The time_mode_list and -vectorfree_modes options are mutually exclusive.

object_list

Specifies a list of PG pins on the instances. When multiple macro model instances are specified, they will be assigned with the same mode sequence.

DESCRIPTION

The command specifies mode sequence related options for analyzing dynamic current and voltage effects of the specified macro models. The options include the pair of time and operation mode, the associated clocks, or the user-specified mode sequence file for both vector-based and vector free rail analysis. You can also specify the operation modes specifically for vectorfree rail analysis only.

For vector-based rail analysis, PrimeRail applies the corresponding current waveforms from the mode at the specified time. For vector free rail analysis, those time points are the triggering clock edges.

EXAMPLES

The following example applies a mode sequence to the PG pin inst4/VDD:

```
pr_shell> report_units
...
Time_unit                   : 1e-09 Second
...
pr_shell> set_rail_macro_mode_sequence {0 Read 1 Write} inst4/VDD
1
...
pr_shell> set_rail_macro_mode_sequence -vectorfree_modes {Read Write} inst4/VDD
1
...
pr_shell> set_rail_macro_mode_sequence -vectorfree_modes {Read Write} inst4/VDD -clock CLK
1
...
pr_shell> set_rail_macro_mode_sequence -vectorfree_modes -import mode_only.seq
1
...
```

SEE ALSO

```
rail_attributes(2)
remove_rail_macro_mode_sequence(2)
update_currents(2)
```

set_rail_macro_model_cells

Specify the attributes of hard macro cells which have macro model.

SYNTAX

```
status set_rail_current

   object_list
   [-internal_net_state ON|OFF]
```

Data Types

ARGUMENTS

object_list

A list of cells which to set attribute.

-internal_net_state *ON|OFF*

Specify the state of internal nets in macro cells. It changes the attribute *is_internal_net_state_on* of macro cells. Default value is OFF state. This option should be set before **update_current** command

DESCRIPTION

The command can change the attribute of macro cells with rail macro model.

Specify the state of internal nets in macro cells. It changes the attribute *is_internal_net_state_on* of macro cells. Default value is OFF state. ON means the internal and external supply nets are connected together. OFF state means the internal net is disconnected with the external net and the net is floating.

EXAMPLES

The following example shows how to change the internal net state of macro cells.

```
pr_shell> set_rail_macro_model_cells [ get_cell mem64x64 ] -internal_net_state OFF
pr_shell> get_attribute [ get_cell mem64x64 ] is_macro_internal_net_on
```

SEE ALSO

create_rail_macro_model(2)
report_rail_macro_model(2)

set_rail_power

Assigns total power to the PG pins of cells to be used in rail analysis.

SYNTAX

```
status set_rail_power
   [power_value]
   [object_list]
   [-supply_net supply_net]
   [-import file_name]
   [-reuse]
```

Data Types

```
float power_value
list object_list
list supply_net
```

string *file_name*

ARGUMENTS

power_value

Defines the power (in Watts) to be assigned to the specified objects.

object_list

Specifies a list of cells, lib_cells or PG pins on which to set rail power.

-supply_net *supply_net*

Identifies a list of supply nets attached to the specific PG pins on which to apply the power. This is an indirect way to get the power onto the desired PG pins that can accommodate different PG pin names across the objects defined in the *object_list*. If no *supply_net* is specified for a given cell or lib_cell, the power value is evenly divided across all PG pins of the object. This option is applied only to the *object_list* items that are cells or lib_cells.

-import *file_name*

Specifies a list of cells, lib_cells or PG pins, and their power values in a given format.

-reuse

Indicates to read in **set_rail_power** from a previous session. User assigned power from **set_rail_power** commands in a previous session is saved to RAIL_DATABASE after a successful run of **calculate_rail_voltage** from the last session.

This option is mutually exclusive with other options in the command.

DESCRIPTION

The **set_rail_power** command assigns power values to PG pins. The PG pins can be directly specified, or indirectly specified by referencing their associated cell instances, lib_cell references or attached supply nets. The command returns 1 if successful, and 0 if unsuccessful.

This command is generally used to provide power information for cases where no correct total power is determined for rail analysis, like a black box that lacks a detailed power model. It can also be used to modify a calculated power number for any cell. The power value assigned by this command always overrides the power value calculated during **update_currents**, irrespective of their execution order. This means that **set_rail_power** has higher precedence than **update_currents**.

When the power is assigned to a power pin, the power for the ground pin will be derived from the power assigned to the power pin by using library PG pin mapping. This ensures that the power on the PG pins of a cell are consistent with respect to each other, such that the total power on a cell's power PG pins is equal to the total power on the corresponding ground PG pins. In this case, the power derived and assigned to the ground pin is considered as the indirectly assigned power to the ground pin.

When the power is assigned to a power pin, static current will also be calculated based on the assigned power and its supply voltage. At the same time, static current is derived for the mapped ground pin. This ensures that the current on the power pin is consistent with the current on its mapping ground pin.

When the power is assigned to a ground pin, the assigned power will first be compared against the power derived from its mapped power pins through previous commands **set_rail_power**. If those two power values are the same, there is no change on the derived power and current values on the ground pin. If those two values are different, the new power value will overwrite the derived power. A new static current will be calculated using the first mapped power pin supply voltage, and this current will overwrite the derived current. In this case, power and current consistency between power and ground pins may be broken, and it is user's responsibility to make them consistent.

If a PG pin is not specified with this command, either directly or by specifying the attached supply net, then the power is divided equally across all the PG pins of the specified cell. If a list or collection of cells are specified, the power value is applied to each cell. When power is applied to lib_cell, all the cells of the lib_cell will have that power applied.

The power value directly assigned to the PG pin has higher precedence than the power value indirectly assigned to the PG pins by either specifying cell instances or lib_cells or mapping from the power pin to the ground pin. A warning message will be issued if the user tries to assign power to the PG pin indirectly, but other source of power information with higher power precedence was already set on the same PG pin.

When a **set_rail_power** command re-assigns power to the same object, the new assignment overrides the previous one. Note that if the earlier assignment does not specify a specific PG pin and the power is divided between two or more pins, and then a later command applies power to a specific PG pin on that same instance, the power assigned to the other PG pins is not affected.

For the file specified in option *-import*, each line of file file_name takes one of the following formats:

CELL cell_instance_name power_value [pattern_match]

PG_PIN cell_instance_name pg_pin_name power_value [pattern_match]

LIB_CELL lib_cell_name power_value [pattern_match]

They correspond to assigning power to cells, PG pins, and lib_cells, described above.

The pattern match indicator [pattern_match] is optional and needed only if the cell_instance_name, pg_pin_name, or lib_cell_name has patterns in its name. The users can simply write "P", "p", "pattern", "pattern_match", or "patternMatch" for the pattern match indicator.

The supported patterns are .*, ?, and [ ]. For example, the following line assigns power 0.5 to all the cells in the design

CELL .* 0.5 P

The following line assigns power 0.8 to the vdd pins of all the cells under macro SUB1

PG_PIN SUB1/.* vdd 0.8 pattern

The following line assigns power 1.0 to the vdd pin of cell SUB1/inv

PG_PIN SUB1/inv vdd 1.0

Note that, if there is no pattern in the name, that is, the exact name is given, it is better for the user not to specify the pattern match indicator, since handling pattern match takes a longer time than the exact name. For example, the following line assigns power 1.5 to cell instance SUB2/inv

CELL SUB2/inv 1.5

When the **set_rail_power** command run is complete, the PG pin *total_power* attribute for the affected PG pins reflects the value of the assigned rail power.

The user can also use the **report_rail_power** command to report the calculated power (0.0 if the **update_currents** command is not performed yet) and the assigned power for every PG pin. The power assigned by the **set_rail_power** command can be removed by the **remove_rail_power** command.

After the power is set, the attribute **has_annotated_power** is set to true on the cells and on the pg_pins of the cells for which the power was set. The **total_power** attribute on the respective cells and pg_pins reflects the new power value.

EXAMPLES

The following example sets the power for the vdd pin of a specific blackbox instance:

```
pr_shell> set_rail_power .01 U153/vdd
```

The following example sets the power for all instances of a specific reference or master cell:

```
pr_shell> set_rail_power .05 [get_lib_cell typical_lib/bufx6]
```

The following command sets the power for all PG pins named vdd2:

```
pr_shell> set_rail_power 0.25 [get_pg_pins */vdd2]
```

The following command sets the power for all the PG pins that connect to supply net VDD:

```
pr_shell> set_rail_power  0.25 -supply_net VDD [get_cells -hier]
```

The following command sets the power for all the PG pins specified by file power_file.txt:

```
pr_shell> set_rail_power  -import  power_file.txt
```

In the following example, a cell INV63 has one ground pin VSS, and two power pins VDD and VCC. VDD supply voltage is 5V, and VCC supply voltage is 1.5V. VSS is mapped to both VDD and VCC power pins. User sets 0.02W

to VDD pin first, and then sets 0.0015W on VCC pin. User does not set any power for ground pin VSS. The VSS power and current will be derived from its mapping power pins VDD and VCC. Thase derived VSS power is 0.0215W, and the derived VSS current is 0.005A.

```
pr_shell>set_rail_power 0.02 [get_pg_pin INV64/VDD]
pr_shell>set_rail_power 0.0015 [get_pg_pin INV64/VCC]
pr_shell>get_attribute [get_pg_pin INV64/VDD] static_current
0.004
pr_shell>get_attribute [get_pg_pin INV64/VCC] static_current
0.001
pr_shell>get_attribute [get_pg_pin INV64/VSS] total_power
0.0215
pr_shell>get_attribute [get_pg_pin INV64/VSS] static_current
0.005
```

Following previous example, if user explicitly sets power again on the VSS pin, but the power is the same as the derived one, there is no change on the ground pin power and current values.

```
pr_shell>set_rail_power 0.0215 [get_pg_pin INV64/VSS]
pr_shell>get_attribute [get_pg_pin INV64/VSS] total_power
0.0215
pr_shell>get_attribute [get_pg_pin INV64/VSS] static_current
0.005
```

Following previous example, if user explicitly sets a different power on the VSS pin, the ground pin power will be updated with the new power number, and static current is also re-calculated using the first mapped power pin VDD supply voltage, which is 5V.

```
pr_shell>set_rail_power 0.05 [get_pg_pin INV64/VSS]
pr_shell>get_attribute [get_pg_pin INV64/VSS] total_power
0.05
pr_shell>get_attribute [get_pg_pin INV64/VSS] static_current
0.01
```

SEE ALSO

rail_attributes(2)
remove_rail_power(2)
report_rail_power(2)

set_related_supply_net

Associates an external supply net to the port of the design.

SYNTAX

```
status set_related_supply_net
   [-object_list ports]
   [-ground ground_net]
   [-power power_net]
   [-reset]
   [power_net]
```

Data Types

```
ports        list
ground_net   string
power_net    string
```

ARGUMENTS

-object_list *ports*

Specifies the list of ports (pins) to be associated with power/ground nets or that must be reset. If this option is not specified, all the ports are considered.

-ground *ground_net*

Specifies the ground net.

-power *power_net*

Specifies the power net.

-reset

Resets the related power and ground nets of the ports specified with the **-object_list** option. This cannot be used in conjunction with either the **-power** or **-ground** option.

power_net

Specifies the power net. This is deprecated and supported for backward compatibility only. New scripts should specify the power net using the **-power** option.

DESCRIPTION

This command is used to associate supply nets with design ports. The tool uses this information for constraining and checking design. The level shifter insertion tool also uses supply nets to determine if level shifter is required between a port and the logic connected to the port.

Setting related supply nets on pins is used for level shifter insertion in synthesis and design optimization tools. PrimeTime accepts pins in **-object_list** but does not use such information for its analysis.

If this command is not specified, the tool assumes that ports are externally connected to the primary supply net of the domain of the top design.

EXAMPLES

The following example sets the *VDD_EXT* power net and *VSS_EXT* ground net with *INPUT* port.

```
pt_shell> set_related_supply_net -object_list \
[get_ports INPUT] -power VDD_EXT -ground VSS_EXT
```

This example resets the power net and ground net on all ports.

```
pt_shell> set_related_supply_net -reset
```

SEE ALSO

create_supply_net(2)
get_supply_nets(2)
get_ports(2)

set_resistance

Sets the **ba_net_resistance** attribute with a resistance value on specified nets.

SYNTAX

```
int set_resistance
   [-min]
   [-max]
   resistance_value
   object_list
```

Data Types

resistance_value	float
object_list	list

ARGUMENTS

-min

Applies only for designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* option is the minimum resistance.

-max

Applies only for designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* option is the maximum resistance.

resistance_value

Specifies the resistance value for nets in the *object_list* option in units consistent with those used by the technology library.

object_list

A list of nets on which the *resistance_value* option is set.

DESCRIPTION

Sets the **ba_net_resistance** attribute, which specifies the back-annotation of resistance values on nets in the current design. If the current design is hierarchical, you must link it using the **link** command. The specified *resistance_value* option overrides the internally estimated net resistance value and also overrides any net resistance set by the **read_parasitics** command.

You can also use the **set_resistance** command for nets at lower levels of the design hierarchy. You can specify these nets as "BLOCK1/BLOCK2/NET_NAME".

To view resistance values, use the **report_net** command.

To remove a resistance value, use the **remove_resistance** command. To reset all back-annotated resistance values in a design, use the **reset_design** command.

EXAMPLES

The following example sets a resistance of 200 units to nets "a" and "b."

```
pt_shell> set_resistance 200 {a,b}
```

In the following example, a resistance of 300 units is set on net "U1/U2/NET3."

```
pt_shell> set_resistance 300 U1/U2/NET3
```

In the following example, the back-annotated resistance on net "U1/U2/NET3" is removed.

```
pt_shell> remove_resistance {get_nets U1/U2/NET3}
```

SEE ALSO

```
find(2)
link(2)
remove_resistance(2)
report_net(2)
report_timing(2)
reset_design(2)
set_drive(2)
```

set_retention

Defines the UPF retention strategy for the power domains in the design.

SYNTAX

```
status set_retention
   -domain domain_name
   [-no_retention]
   [-retention_power_net supply_net_name]
   [-retention_ground_net supply_net_name]
   [-retention_supply_set supply_set_name]
   [-save_signal save_signal_and_save_sense]
   [-restore_signal restore_signal_and_restore_sense]
   [-elements object_list]
   retention_name
```

Data Types

```
domain_name              string
supply_net_name          string
supply_set_name          string
save_signal_and_save_sense    string
restore_signal_and_restore_sense     string
object_list              list
retention_name           string
```

ARGUMENTS

-domain *domain_name*

Specifies the power domain name that this UPF retention strategy is applied to.

-no_retention

Specifies that the specified objects in the power domain does not have retention.

-retention_power_net *supply_net_name*

Specifies the retention power net for the retention cells that are under this UPF retention strategy.

-retention_ground_net *supply_net_name*

Specifies the retention ground net for the retention cells that are under this UPF retention strategy.

-retention_supply_set *supply_set_name*

Specifies the supply set whose power and ground function associations need to be used as the retention power and retention ground nets respectively. This argument is mutually exclusive with **-retention_power_net** and

-retention_ground_net.

-save_signal ***save_signal_and_save_sense***

Specifies the save signal net and the save signal sense for the retention cells under this retention strategy. Use the following syntax: {*save_signal* high|low}.

-restore_signal ***restore_signal_and_restore_sense***

Specifies the restore signal net and the restore sense for the retention cells under this retention strategy. Use the following syntax: {*restore_signal* high|low}.

-elements ***object_list***

Specifies the objects to which this UPF retention strategy is applied. The objects can be hierarchical cells, leaf cells, hdl blocks, nets.

retention_name

Specifies the UPF retention strategy name. The retention strategy name should be unique within the specified power domain.

DESCRIPTION

This command defines the UPF retention strategy on the specified power domain under which to map the unmapped sequential cells to retention cells.

PrimeTime uses the **set_retention** command to build virtual power ground connectivity. The command creates explicit connection for retention(backup) power and ground nets to power and ground pins of retention cells. The **-no_retention** option on specific elements takes precedence over domain-wide strategy.

If the **-elements** option is not specified, the retention strategy is applied to all of the unmapped sequential cells under the power domain. If the **-elements** option is specified, the UPF retention strategy is applied to all the unmapped sequential cells that are under the objects from **-elements**.

If neither the **-retention_supply_set** argument nor **-retention_power_net/-retention_ground_net** is specified, the retention power and ground nets are automatically set to the power and ground functions of the **default_retention** supply set handle of the power domain, for which the strategy is being defined.

For a list of UPF commands, use this command:

```
prompt>  help upf
```

EXAMPLES

The following example shows how to define a UPF retention strategy in the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_retention retention_1 -domain PD1 \
          -retention_power_net PN1 \
          -retention_ground_net GN1 \
```

In the following example, it shows how to define a UPF retention strategy in the objects under the specified power

domain.

```
prompt> set_retention retention_2 -domain PD1 \
        -retention_power_net PN1 \
        -retention_ground_net GN1 \
        -elements shutdowm_inst/mid_inst \
```

The following example shows how to define a UPF retention strategy using a supply set.

```
prompt> set_retention retention_2 -domain PD1 \
        -retention_supply_set ret_supply_set \
        -elements shutdown_inst/mid_inst/R_reg
```

The following example shows how a UPF retention strategy can be defined without using a supply set or supply nets.

```
prompt> set_retention retention_2 -domain PD1 \
        -elements shutdown_inst/mid_inst/R_reg
```

SEE ALSO

```
create_power_domain(2)
create_supply_net(2)
create_supply_set(2)
report_power_pin_info(2)
set_retention_control(2)
```

set_retention_control

Defines the UPF retention control signals for the defined UPF retention strategy.

SYNTAX

```
status set_retention_control
   -domain power_domain
   -save_signal save_signal_and_save_sense
   -restore_signal restore_signal_and_restore_sense
   retention_strategy_name
```

Data Types

```
power_domain                        string
save_signal_and_save_sense          string
restore_signal_and_restore_sense    string
retention_strategy_name             string
```

ARGUMENTS

-domain *power_domain*

Specifies the power domain name that this UPF retention strategy that is applied to.

-save_signal *save_signal_and_save_sense*

Specifies the save signal net and the save signal sense for the retention cells under this retention strategy. Use the following syntax: {*save_signal* high|low}.

-restore_signal *restore_signal_and_restore_sense*

Specifies the restore signal net and the restore sense for the retention cells under this retention strategy. Use the following syntax: {*restore_signal* high|low}.

retention_strategy_name

Specifies the UPF retention strategy name. The retention strategy should have already been defined using **set_retention** command

DESCRIPTION

This command defines the UPF retention control signals and control signal sense for this retention strategy of this power domain. The specified control signals are applied to the retention cells under the associated retention

strategy.

PrimeTime uses additional information about the retention strategy provided by this command to properly match retention strategies to individual leaf cells. Retention strategy is used for deriving PG connectivity from UPF description. When multiple retention strategies match a retention cell, the last one entered is used.

EXAMPLES

The following example shows how to define the retention control signals on the defined UPF retention strategy *retention_1* of the power domain *PD1*.

```
prompt> set_retention_control retention_1 \
         -domain PD1 \
         -save_signal {save_net high} \
         -restore_signal {restore_net low}
```

The following example shows how to find which retention strategy was used on which retention register to derive backup PG connectivity:

```
foreach_in_collection c [sort_collection
  [get_cells -hier * -filter "upf_retention_strategy != {}"] full_name] {
    echo [format
      {  Cell  %-8s  retention %s} [get_attribute $c full_name]
        [get_attribute $c upf_retention_strategy]]
}
```

SEE ALSO

set_retention(2)

set_rtl_to_gate_name

Sets the name mapping between the RTL and gate-level objects. Use this mapping to read the RTL backward SAIF file or VCD file for power estimation.

SYNTAX

```
integer set_rtl_to_gate_name
   -rtl rtl_name
   -gate gate_name
   -substitute substitute
   -inverted
```

Data Types

```
rtl_name                string
gate_name               string
substitute              string
```

ARGUMENTS

-rtl *rtl_name*

This option provides the name of a RTL object, whose name has changed after synthesis. The RTL object appears in the RTL backward SAIF or VCD file.

-gate *gate_name*

This option provides the name of gate-level object, for the corresponding RTL object. The name of the RTL object in the RTL backward SAIF or VCD file might change after synthesis. The *gate_name* is the new gate-level name of the RTL object after synthesis. This RTL object can be mapped to multiple gate-level objects due to replication.

-substitute *substitute*

This option directs the tool to substitute all the *from_string* occurrences in the RTL VCD or SAIF file with the *to_string*. Specify as *-substitute {from_string to_string}*.

-inverted

This option specifies that the signal in the RTL activity file gets inverted when mapped to the gate-level design. You can use this option only when the object specified with the *-gate* option is a net or a pin.

DESCRIPTION

After synthesis, the names of the RTL objects might change due to which the **read_saif** or **read_vcd** command is not able to map the names present in the RTL SAIF or VCD file to the gate-level objects. Due to this, many of the nets, ports, and pins might not have the user-defined switching activity values, resulting in inaccurate power numbers.

To avoid this situation, the **set_rtl_to_gate_command** command allows you to set the mapping between the RTL and gate-level names after you have completed the RTL simulation only. You can set the RTL and gate-level names by using the *-rtl* and *-gate* options, respectively.

During the **read_saif** and **read_vcd** command, if the tool is unable to find the gate-level object matching the RTL object name, it looks for the corresponding gate-level name; however, if the gate-level name is provided using the **set_rtl_to_gate_command** command, the tool tries to find the corresponding gate-level object.

Note: Specify the full name of the object with respect to the current instance to the *gate_name* argument. Similarly, specify the the full name as it appears in the backward RTL SAIF or VCD file to the *rtl_name* argument.

When you use the *-substitute* option, any signal name in the RTL VCD or SAIF is scanned only one time. The substitutions are not applied recursively.

EXAMPLES

The following example provides the mapping between RTL and gate-level objects.

```
pt_shell> set_rtl_to_gate_name -rtl reg_i[1] -gate reg_i_1
```

The following example provides the mapping for an RTL level signal to a gate-level pin, and instructs the tool to invert the signal found in the RTL activity file.

```
pt_shell> set_rtl_to_gate_name -rtl test_signal \
             -gate test_signal_reg/Q  -inverted
```

SEE ALSO

```
unset_rtl_to_gate_name(2)
reset_rtl_to_gate_name(2)
report_switching_activity(2)
read_saif(2)
reset_switching_activity(2)
set_switching_activity(2)
get_switching_activity(2)
report_power(2)
report_activity_file_check(2)
```

set_scope

Specify the current UPF scope. Return the current UPF scope prior to the execution of this command as a full path string relative to the current design top if successful and null string if it fails.

SYNTAX

```
string set_scope
   [instance]
```

Data Types

instance	string

ARGUMENTS

instance

Specifies the working instance in pt_shell. If the *instance* option is not specified, the focus returns to the top level of the current design. If *instance* is ".", pt_shell returns to the working instance. If *instance* is "..", the context is moved up one level in the instance hierarchy. If *instance* begins with "/", pt_shell returns to the working instance of the design whose name is after the "/". More complex examples of *instance* arguments are described below in EXAMPLES.

DESCRIPTION

This command sets the UPF scope in pt_shell. Functionally, this command is the subset of the **current_instance** command, but with different return value.

- If no *instance* argument is specified, the UPF scope of pt_shell is returned to the top level of the hierarchy.
- If *instance* is ".", the UPF scope is returned and no change is made.
- If *instance* is "..", the UPF scope is moved up one level in the design hierarchy.
- If *instance* is a valid cell at the current level of hierarchy, the UPF scope is moved down to that level of the design hierarchy.
- Multiple levels of hierarchy can be traversed in a single call to **set_scope** by separating multiple cell names with slashes.

  For example, the **set_scope U1/U2** command sets the UPF scope down two levels of hierarchy if both cells exist at the current levels in the design hierarchy.

- The ".." directive can also be nested in complex *instance* arguments.

  For example, the **set_scope "../../MY_INST"** command attempts to move the context up two levels of hierarchy, then down one level to the "MY_INST" cell.

NOTE: The **set_scope** command does not work on leaf cells. If you attempt to run this command on a leaf cell, an error occurs.

SEE ALSO

```
current_instance(2)
current_design(2)
```

set_sense

Specifies unateness propagating forward for pins with respect to clock source.

SYNTAX

```
status set_sense
   [-type clock | data]
   [-non_unate]
   [-rise]
   [-fall]
   [-clock_fall]
   [-clock_rise]
   [-primary]
   [-generated]
   [-positive]
   [-negative]
   [-stop_propagation]
   [-clock_leaf]
   [-pulse pulse_type]
   [-clocks clock_list]
   object_list
```

Data Types

```
pulse_type     string
clock_list     list
object_list     list
```

ARGUMENTS

-type clock | data

Specifies whether the sense is applied to clock or data networks. If you do not specify this option, the default is **clock**. If you specify **data**, you must also use the **-stop_propagation** and **-clocks** options.

-non_unate

Launches both the positive-unate sense and the negative-unate sense of the clock from the clock source. If you use the **-non_unate** option, you must also use the **-clocks** option. Please note that this option is only valid for clock source pins/ports.

-rise

Specifies rising clock sense. Please note that this option is only valid for clock source pins/ports.

-fall

Specifies falling clock sense. Please note that this option is only valid for clock source pins/ports.

-clock_fall

Specifies that the signal is coming from the falling edge of the clock. If not specified, the command uses the falling edge for **-fall**. Please note that this option is only valid for clock definition points.

-clock_rise

Specifies that the signal is coming from the rising edge of the clock. If not specified, the command uses the rising edge for **-rise**. Please note that this option is only valid for clock definition points.

-primary

Specifies primary clock signal type, primary is the default. Please note that this option is only valid for clock definition points.

-generated

Specifies generated clock signal type. Please note that this option is only valid for clock definition points.

-positive

Applies positive unateness to all pins in the *object_list* with respect to clock source. If you use the **-positive** option, you cannot use the **-negative** or **-pulse** options.

-negative

Applies negative unateness to all pins in the *object_list* with respect to clock source. If you use the **-negative** option, you cannot use the **-positive** or **-pulse** options.

-stop_propagation

Stops the propagation of specified clocks in the *clock_list* from the specified pins or cell timing arcs in the *object_list*. Only a clock used as a clock is stopped if you specify the **-type clock** option. To stop propagation for both clock-as-clock and clock-as-data, you need to use a **set_sense -type clock** command with a **set_sense -type data** command. You cannot use the **-stop_propagation** option with the **-positive**, **-negative**, or **-pulse** options.

-clock_leaf

Stops the propagation of specified clocks in the *clock_list* from the specified pins in the *object_list*. Also, it specifies the pin as a clock consumer, so that the clock branches reaching this pin will be considered as clock consumer as well. In other words, the attribute **"is_clock_used_as_clock"** will be true for all the pins of the clock branches that reach the clock leaf pin. You cannot use the **-clock_leaf** option with the **-positive**, **-negative**, **-stop_propagation** or **-pulse** options.

-pulse *pulse_type*

Specifies the type of pulse clock applied to all pins in the *object_list* with respect to clock source. The possible values for the *pulse_type* argument are

- **rise_triggered_high_pulse**
- **fall_triggered_high_pulse**
- **rise_triggered_low_pulse**
- **fall_triggered_low_pulse**

If you use the **-pulse** option, you cannot use the **-positive** or **-negative** options.

-clocks *clock_list*

Specifies a list of clock objects to be applied with the given unateness that is placed on all pin objects in the *object_list*. If you do not specify the **-clocks** option, all clocks passing through the given pin objects are considered.

object_list

List of pins, ports, or cell timing arcs with specified unateness to propagate. The timing arcs object can be used with the **-stop_propagation** option only.

DESCRIPTION

This command modifies the waveform of the clock in two distinct ways:

1. You can use this command to restrict unateness at pin (to positive or negative unate) with respect to clock source. However, the specified unateness only applies within the non-unate clock network. In this case, user-defined sense propagates forward from the given pins.

2. You can specify the waveform of the clock at its definition point. This option is only valid for the clock sources of the clock. You can specify the waveform to be launched from the clock source as positive, negative or non-unate. You can also specify whether the clock is only used for source latency computation of the generated clocks. This can be specified by the **-generated** option.

If you use the **-clocks** option, only the specified clock domains are applied. Otherwise, all clocks passing through the given pin objects are considered.

PrimeTime issues warning messages if the specified sense on given pins cannot be respected in case there is predefined unateness for given pins. A hierarchical pin is not supported.

In addition, you can completely stop propagation of certain clocks in clock networks or data networks by using the **-stop_propagation** option along with the **-type** option.

To undo the **set_sense** command, use the **remove_sense** command.

EXAMPLES

The following example specifies a positive unateness for the XOR/Z pin with respect to the CLK1 clock.

```
pt_shell> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies negative unateness for the MUX/Z pin for all clocks.

```
pt_shell> set_sense -negative MUX/Z
```

The following example specifies a positive unate clock sense on primary clock CLK1 using a combination of -fall, -rise, -clock_fall, -clock_rise.

```
pt_shell> set_sense -primary -rise -clock_rise -clocks CLK1 inv2/Y
pt_shell> set_sense -primary -fall -clock_fall -clocks CLK1 inv2/Y
```

The following example specifies a positive unate waveform for the clock and a negative unate waveform for the clock when it is used for source latency computation.

```
pt_shell> set_sense -primary -fall -clock_fall -clocks CLK1 inv2/Y
pt_shell> set_sense -primary -rise -clock_rise -clocks CLK1 inv2/Y
```

```
pt_shell> set_sense -generated -fall -clock_rise -clocks CLK1 inv2/Y
pt_shell> set_sense -generated -rise -clock_fall -clocks CLK1 inv2/Y
```

SEE ALSO

remove_sense(2)

set_setup_hold_pessimism_reduction

Sets the optimization constraints for setup-hold pessimism reduction.

SYNTAX

```
status set_setup_hold_pessimism_reduction
   -mode mode_name
   [-setup_cutoff setup_cutoff_slack]
   [-hold_cutoff hold_cutoff_slack]
```

Data Types

```
mode_name                 string
setup_cutoff_slack        float
hold_cutoff_slack        float
```

ARGUMENTS

-mode *mode_name*

Specifies the mode of the setup-hold pessimism reduction (SHPR). The valid values are: **total**, **setup**, or **hold**.

-setup_cutoff *setup_cutoff_slack*

Specifies the cutoff slack of the MAX path for SHPR optimization. Must be a float value. The default is negative INFINITY.

-hold_cutoff *hold_cutoff_slack*

Specifies the cutoff slack of the MIN path for SHPR optimization. Must be a float value. The default is negative INFINITY.

DESCRIPTION

This command sets the optimization constraints for setup-hold pessimism reduction (SHPR). The valid modes are: *total*, *setup*, and *hold*. The different modes control how setup slack can be traded off for hold slack or vice versa.

In *setup* mode (also known as setup-preferred mode), the goal is to prefer positive setup slack at the expense of hold slack. This mode might be useful for performance-critical designs, knowing that the hold slacks can be improved later through strategic buffer insertion.

In *hold* mode (also known as hold-preferred mode), the goal is to prefer positive hold slack at the expense of setup slack. This mode might be useful in the late stages of timing closure, when it is desirable to see if small hold

violations can be resolved without any design changes.

The *total* mode trades off setup and hold slack in a way that minimizes the sum of the negative rise/fall setup/hold slack. This mode results in the smallest overall set of setup and hold timing violations. This is the default mode of SHPR.

Of course, it might not be desirable to introduce a large violation in the sacrificial slack type to gain only a small improvement in the preferred slack type. To keep the trade-off reasonable, the **-setup_cutoff** and **-hold_cutoff** options have been provided. In setup-preferred mode, hold violations are made no worse than the **-hold_cutoff** value to improve setup. In hold-preferred mode, setup violations are made no worse than the **-setup_cutoff** value to improve hold. The default for both cutoff options is negative infinity, which allows any amount of slack worsening in the sacrificial slack to improve the preferred slack.

In both modes, the preferred slack type has improved only as far as it takes to reach zero slack, and no further.

The slacks being analyzed above are the slacks of timing paths ending at sequential cell data pins which have SHPR constraint information.

EXAMPLES

The following example sets the optimization mode to **total** while setting the *cutoff_slack* of the **-setup_cutoff** and **-hold_cutoff** to 0 and -0.03ns, respectively.

```
pt_shell> set_setup_hold_pessimism_reduction -mode total \
                        -setup_cutoff 0 -hold_cutoff -0.03
1
```

The following example sets the optimization mode to **setup** while setting the *cutoff_slack* of **-hold_cutoff** to 0.03ns, respectively.

```
pt_shell> set_setup_hold_pessimism_reduction -mode setup \
                        -hold_cutoff 0.03
1
```

The following example sets the optimization mode to **hold** while setting the *cutoff_slack* of **-setup_cutoff** to -0.05ns, respectively.

```
pt_shell> set_setup_hold_pessimism_reduction -mode hold \
                        -setup_cutoff 0.03
1
```

SEE ALSO

remove_setup_hold_pessimism_reduction(2)

set_si_aggressor_exclusion

Sets the given nets to be exclusive while switching in the given direction, when they are aggressors to the same victim net.

SYNTAX

```
integer set_si_aggressor_exclusion
   [-number_of_active_aggressors n]
   [-rise]
   [-fall]
   anets
```

Data Types

```
anets        list
n            integer
```

ARGUMENTS

-number_of_active_aggressors *n*

Specifies the maximum number of aggressors among the given list of exclusive aggressors that can be active at a given instant. This option is used to set n-hot or n-cold behavior, where only the *n* nets are active(hot) or quiet(cold) at a given instant. If this option is not specified, the default value of *n=1* is used; i.e only one of the given exclusive aggressors is considered active at a given time.

-rise

Specifies the aggressor nets to be exclusive while switching in the rise direction. If neither the *-rise* nor the *-fall* options are specified, the aggressor *anets* nets are considered exclusive in the directions of both the *-rise* and *-fall* options.

-fall

Specifies the aggressor nets to be exclusive while switching in the fall direction. If neither the *-rise* nor *-fall* options are specified, the aggressor *anets* net are considered exclusive in in the directions of both the *-rise* and *-fall*.

anets

Specifies the list of nets which are exclusive when they are aggressors to the same victim net.

DESCRIPTION

The **set_si_aggressor_exclusion** command sets the aggressor nets to exclusive while switching in the given direction at the same time. Among all combinations of active aggressors allowed by the **-number_of_active_aggressors option, only the combination which produces the worst alignment are considered for crosstalk analysis.**

Only the aggressor nets specified in the *anets* list are considered exclusive. If a victim net is specified in the *anets* list, it is not considered exclusive with any of its aggressor nets.

To view which aggressor nets were considered active and which aggressor nets were set to quiet, use the **report_delay_calculation** or **report_noise_calculation** commands on the victim net. The quiet aggressor nets are reported to be screened due to aggressor exclusion.

These exclusive commands are independent of parasitics, so they are applied even before reading in parasitics.

Note that application of exclusive aggressors cannot be done incrementally; next update_timing and update_noise would be a full update.

The command **report_si_aggressor_exclusion** is used to check which groups are set by the **set_si_aggressor_exclusion** command.

EXAMPLES

The following example shows how the nets *SCAN_LOGIC** are set to be exclusive while switching in the rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
```

The following example shows how the nets *LCO_LOGIC** are set to be exclusive while switching in the fall direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets LCO_LOGIC*] -fall
1
```

The following example shows how only 3 of the nets *LOGIC_BUS** considered to be active while switching in the rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets LOGIC_BUS*] -rise
-number_of_active_aggressors 3
1
```

SEE ALSO

```
remove_si_aggressor_exclusion(2)
report_si_aggressor_exclusion(2)
report_delay_calculation(2)
report_noise_calculation(2)
si_analysis_logical_correlation_mode(3)
set_si_delay_analysis(2)
set_si_noise_analysis(2)
```

set_si_delay_analysis

Sets coupling information on nets for crosstalk analysis.

SYNTAX

```
integer set_si_delay_analysis
   [-ignore_arrival inets]
   [-exclude]
   [-victims vnets]
   [-aggressors anets]
   [-rise]
   [-fall]
   [-min]
   [-max]
```

Data Types

```
inets       list
vnets       list
anets       list
```

ARGUMENTS

-ignore_arrival *inets*

Specifies a list of nets to be analyzed as infinite window. You cannot use this option with the *-exclude*, *-victims*, or *-aggressors* options. The physical exclusivity analysis for SI is considered for these nets when si_xtalk_use_physical_exclusivity_on_ignore_arrival_nets variable is set to true.

-exclude

Indicates that nets specified as a *vnets* or *anets* variable are to be excluded from the crosstalk analysis as victim nets or aggressor nets, respectively. You cannot use this option with the *-ignore_arrival* option. When both the *-victims vnets* and *-aggressors anets* options are applied, all cross capacitances between the *vnets* and *anets* variables are excluded, when *vnets* are victims and *anets* are aggressors.

-victims *vnets*

Specifies the list of nets on which the *-exclude* option information is applied as a victim. You cannot use this option with the *-ignore_arrival* option. If you use the *-victims* option, you must use the *-exclude* option. When used with the *-aggressors* option, the *-victims* option excludes the cross capacitances between the victim nets (*vnets*) and the aggressor nets (*anets*).

-aggressors *anets*

The list of nets on which the *-exclude* option information is applied as an aggressor. You cannot use this option with the *-ignore_arrival* option. If you use the *-aggressors* option, you must use the *-exclude* option. When used

with the *-victims* option, the *-aggressors* option excludes the cross capacitances between the victim nets (*vnets*) and the aggressor nets (*anets*).

-rise

Excludes a list of nets for victim rising. If you use the *-rise* option, you must use the *-exclude* option.

-fall

Excludes a list of nets for victim falling. If you use the *-fall* option, you must use the *-exclude* option.

-min

Excludes a list of nets for min path analysis. If you use the *-min* option, you must use the *-exclude* option.

-max

Excludes a list of nets for max path analysis. If you use the *-max* option, you must use the *-exclude* option.

DESCRIPTION

Sets coupling information on nets for crosstalk analysis.

The **set_si_delay_analysis** command allows you to set some net as infinite window as both aggressor and victim. This command has no default. To remove the result of this command, use the **remove_si_delay_analysis** command.

The **set_si_delay_analysis** command and the *-exclude* option allows you to select nets to be excluded from crosstalk analysis. By default, all nets with coupling capacitances (non-filtered) are included in crosstalk analysis as both victim and aggressor. This command allows the nets to be excluded as victim (by using the *-victim* option) or aggressor (by using the *-aggressors* option). When you use both the *-victims* and *-aggressors* options, the command excludes the relationship between the specified victim nets (*vnets*) and aggressor nets (*anets*); this is referred to as "pair-wise exclusion."

By using the **set_si_delay_analysis** command with the *-ignore_arrival* option, you can set the *inets* as infinite window. When *inets* is analyzed as victim, the timing relationship between all the nets in this coupling cluster is ignored, and all aggressors become active (even those driven by clock domains which are physically exclusive). When *inets* is analyzed as aggressor the arrival time of *inets* is ignored. The infinite window analysis could make the crosstalk calculation more pessimistic.

The **set_si_delay_analysis** command returns a **1** if successful and a **0** if unsuccessful.

To view the results of this command, use the **report_si_delay_analysis** command.

EXAMPLES

The following example shows that all nets described by *CLK_NET_* * are excluded from crosstalk analysis as victim nets.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET*]
1
```

The following example shows how all nets in a design to be analyzed as infinite window.

```
pt_shell> set_si_delay_analysis -ignore_arrival [get_nets -hier *]
1
```

The following example shows how to ensure that the scan clock described by *SCN_CLK_** is not effecting the main clock network described by *CLK_NET_**, and vice versa.

```
pt_shell> set_si_delay_analysis -exclude -victims \
[get_nets SCN_CLK_*] \
-aggressors [get_nets CLK_NET*]
pt_shell> set_si_delay_analysis -exclude -aggressors \
[get_nets SCN_CLK_*] -victims [get_nets CLK_NET*]
1
```

SEE ALSO

```
read_parasitics(2)
remove_si_delay_analysis(2)
report_si_delay_analysis(2)
si_enable_analysis(3)
```

set_si_delay_disable_statistical

Disables composite aggressor statistical analysis on nets for crosstalk analysis.

SYNTAX

```
int set_si_delay_disable_statistical
   dnets
```

Data Types

```
dnets      list
```

ARGUMENTS

dnets

A list of nets in the current design for which the composite aggressor statistical analysis is disabled.

DESCRIPTION

If selected in composite aggressor group for crosstalk analysis, *dnets* specified by this command are not treated statistically. If they are not selected into composite aggressor group, this command has no effect.

To remove the result of this command, use the **remove_si_delay_disable_statistical** command.

EXAMPLES

The following example shows how to disable net from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_delay_disable_statistical [get_nets LOGIC1]
1
```

SEE ALSO

```
remove_si_delay_disable_statistical(2)
report_si_delay_analysis(2)
```

set_si_noise_analysis

Sets coupling information on nets for noise analysis.

SYNTAX

```
int set_si_noise_analysis
   [-ignore_arrival inets]
   [-exclude]
   [-victims vnets]
   [-aggressors anets]
   [-above]
   [-below]
   [-low]
   [-high]
```

Data Types

inets list *vnets* list *anets* list

ARGUMENTS

-ignore_arrival *inets*

Specifies a list of nets to be set as infinite window. When *inets* are analyzed as victims, all of their aggressors are set with infinite window. When *inets* are analyzed as aggressors, they are set as aggressors with arrival time ignored. You cannot use this option with the *-exclude*, *-victims*, *-aggressors*, *-high*, *-low*, *-above*, or *-below* option.

-exclude

Indicates that nets specified as *vnets* or *anets*) are to be excluded from the noise analysis as victim nets or aggressor nets, respectively. You cannot use this option with the *-ignore_arrival* option. When both the *-victims vnets* and *-aggressors anets* options are applied, the noise between *vnets* and *anets* is not analyzed, when *vnets* are victims and *anets* are aggressors.

-victims *vnets*

Specifies the list of nets on which *-exclude* information is applied as a victim. You cannot use this option with the *-ignore_arrival* option. If you use the *-victims* option, you must use the *-exclude* option. When used with the *-aggressors* option, the *-victims* option excludes the noise analysis between the victim nets (*vnets*) and the aggressor nets (*anets*).

-aggressors *anets*

Indicates the list of nets on which the *-exclude* option information is applied as an aggressor. You cannot use this option with the *-ignore_arrival* option. If you use the *-aggressors* option, you must use the *-exclude* option. When used with the *-victims* option, *-aggressors* excludes the noise analysis between the victim nets (*vnets*) and

the aggressor nets (*anets*).

-above

Excludes a list of nets for victim above rail. If you use the *-above* option, you must use the *-exclude* option.

-below

Excludes a list of nets for victim below rail. If you use the *-below* option, you must use the *-exclude* option.

-low

Excludes a list of nets for low rail noise analysis. If you use the *-low* option, you must use the *-exclude* option.

-high

Excludes a list of nets for high rail noise analysis. If you use the *-high* option, you must use the *-exclude* option.

DESCRIPTION

Sets coupling information on nets for noise analysis.

The **set_si_noise_analysis** command allows you to exclude some net from noise analysis or set some net as infinite window as aggressor. This command has no default. To remove the result of this command, use the **remove_si_noise_analysis** command.

The *-exclude* option of the **set_si_noise_analysis** command allows you to select nets to be excluded from noise analysis. By default, all nets with coupling capacitances (non-filtered) are included in noise analysis as both victim and aggressor. This command allows the nets to be excluded as victim (by using the *-victim* option) and/or aggressor (by using the *-aggressors* option). When you use both the *-victims* and *-aggressors* options, the command excludes the noise analysis between the specified victim nets (*vnets*) and aggressor nets (*anets*); this is referred to as "pair-wise exclusion."

By using the **set_si_noise_analysis** command with the *-ignore_arrival* option, you can set the *inets* as infinite window as aggressor if they are analyzed as aggressors. When *inets* are analyzed as noise victims, all of their aggressors are infinite window.

The **set_si_noise_analysis** command returns a *1* if successful and a *0* if unsuccessful.

To view the results of this command, use the **report_si_noise_analysis** command.

EXAMPLES

The following example shows that all nets described by *CLK_NET_* * are excluded from noise analysis as victim nets.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET*]
1
```

The following example shows that when all nets described by *SOME_LOGIC** are analyzed as aggressors, they are set as infinite window as aggressors in noise analysis.

```
pt_shell> set_si_noise_analysis -ignore_arrival [get_nets SOME_LOGIC*]
1
```

The following example shows how to ensure that the scan clock described by *SCN_CLK_** is not effecting the main clock network described by *CLK_NET_**, and vice versa.

```
pt_shell> set_si_noise_analysis -exclude -victims \
           [get_nets SCN_CLK_*] \
           -aggressors [get_nets CLK_NET*]
pt_shell> set_si_noise_analysis -exclude -aggressors \
           [get_nets SCN_CLK_*] -victims [get_nets CLK_NET*]
1
```

SEE ALSO

```
read_parasitics(2)
remove_si_noise_analysis(2)
report_si_noise_analysis(2)
si_enable_analysis(3)
```

set_si_noise_disable_statistical

Disables composite aggressor statistical analysis on nets for noise analysis.

SYNTAX

```
int set_si_noise_disable_statistical
   dnets
```

Data Types

```
dnets      list
```

ARGUMENTS

dnets

A list of nets in the current design for which the composite aggressor statistical analysis is disabled.

DESCRIPTION

If selected in composite aggressor group for noise analysis, *dnets* specified by this command are not treated statistically. If they are not selected into composite aggressor group, this command has no effect.

To remove the result of this command, use the **remove_si_noise_disable_statistical** command.

EXAMPLES

The following example shows how to disable net from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_noise_disable_statistical [get_nets LOGIC1]
1
```

SEE ALSO

```
remove_si_noise_disable_statistical(2)
report_si_noise_analysis(2)
```

set_spice_options

Sets up configuration options for transistor-level SPICE simulation.

SYNTAX

```
status set_spice_options
   [-spice_exec spice_exec_name]
   [-top_spice_ckt_file file_name]
   [-setup_file file_name]
   [-include_path dir_name]
   [-hdl_path dir_name]
   [-cell_name_mapping cell_name_list]
   [-pin_name_mapping pin_name_list]
   [-extra_options option_string]
```

Data Types

```
string spice_exec_name
string file_name
string dir_name
list cell_name_list
list pin_name_list
string option_string
```

ARGUMENTS

-spice_exec *spice_exec_name*

Specifies the full path of the SPICE or other transistor-level simulator that will be used for transistor-level simulation.

-top_spice_ckt_file *file_name*

Specifies the top-level SPICE sub-circuit netlist file that contains main devices to be simulated during transistor-level simulation. For example, to run simulation on voltage regulators, this file must contain top-level sub-circuit netlists for the voltage regulators.

-setup_file *file_name*

Specifies the SPICE simulation setup file that contains all the needed include files, such as netlist files and model files.

-include_path *dir_name*

Specifies the path of the include files which are not specified with full path in the setup file. Otherwise, deselect this option.

-hdl_path ***dir_name***

Specifies the search path for the Verilog file which is not specified with full path in the setup file. Otherwise, deselect this option.

-cell_name_mapping ***cell_name_list***

Specifies the name of file that describes the cell name mapping between the design and the SPICE sub-circuit netlist. In the mapping file, cell names are listed in the format of cell_in_design: cell_in_netlist. When cell name mapping is not available, all the cells are assumed to have the same names in both the design and the SPICE netlist.

-pin_name_mapping ***pin_name_list***

Specifies the name of the file that describes the pin name mapping between the design and the SPICE sub-circuit netlist. In the mapping file, pin names are listed in format of pin_in_design:pin_in_netlist.

This pin name mapping list applies to all cells that are specified in the cell name mapping file with the -cell_name_mapping option. When no cell name mapping file is available, the tool applies this pin name mapping to all cells in the design. When no pin name mapping is specified, all pins are assumed to have the same name in both the design and the SPICE netlist.

-extra_options ***option_string***

Specifies one or more SPICE command line options other than -include_path and -hdl_path. These options are handed over to the SPICE executable specified by the -spice_exec option.

DESCRIPTION

This command specifies the setup options needed for running transistor-level simulation on cell designs that do not have libraries, such as .db files. For example, you need to simulate a design that has voltage regulator cells but no characterized cell models in the library.

You can execute the command multiple times with the options specified incrementally. If one option is set multiple times, the most recent setting overwrites the earlier one. Note that you need to specify the -cell_name_mapping and -pin_name_mapping options in the same command run. If only one mapping option (either cell or pin) is specified in the command, the tool assumes the other mapping file is empty and no mapping is applied.

To report the SPICE setup options, use the **report_spice_options** command.

EXAMPLES

Assuming the voltage regulator cell master, named VR1, does not exist in the library, the following example specifies the setup options for running SPICE simulation on this voltage regulator cell.

```
pr_shell> set_spice_options \
 -spice_exec "/full/path/spice/exec_name" \
 -top_spice_ckt_file "VR1.net" \
 -setup_file "setup_file_name" \
 -include_path "/dir/for/include/files"
 -extra_options "-mt 4"
1
```

```
pr_shell> report_spice_options
Spice Exec = /full/path/spice/exec_name
Spice TopCktFile = VR1.net
Setup File = setup_file_name
Spice IncPath = /dir/for/include/files
Spice HdlPath = NULL
Spice deck file name for voltage regulator and core co-simulation = Core_VoltageRegulator_cosim.sp
There is no master name mapping and pin name mapping specified.
1

pr_shell> set_spice_options -cell_name_mapping {VR1mw VR1spice} \
 -pin_name_mapping {pin1mw pin1spice pin2mw pin2spice pin3mw pin3spice}
1

pr_shell> report_spice_options
Spice Exec = /full/path/spice/exec_name
Spice TopCktFile = VR1.net
Setup File = setup_file_name
Spice IncPath = /dir/for/include/files
Spice HdlPath = NULL
Spice deck file name for voltage regulator and core co-simulation = Core_VoltageRegulator_cosim.sp
Voltage Regulator master: mw_name=VR1mw, spice_name=VR1spice.
         pin names in MW: pin1mw pin2mw pin3mw
         pin names in spice: pin1spice pin2spice pin3spice
1
```

SEE ALSO

report_spice_options(2)

set_steady_state_resistance

Specifies the steady-state resistance for a library pin or port.

SYNTAX

```
status set_steady_state_resistance
   [-above]
   [-below]
   [-low]
   [-high]
   res_value
   object_list
```

Data Types

```
res_value       float
object_list     list
```

ARGUMENTS

-above

Specifies steady state resistance for above ground or power rail noise analysis region.

-below

Specifies steady state resistance for below ground or power rail noise analysis region.

-low

Specifies steady state resistance for ground rail noise.

-high

Specifies steady state resistance for power rail noise.

res_value

Specifies the steady state resistance value in units of resistance.

object_list

Specifies a list of library pins or ports.

DESCRIPTION

A steady-state driver of a net affects the size of crosstalk bumps on the net due to its loading effects on the net. To accurately calculate the characteristics of crosstalk noise bumps, the tool needs the steady-state I-V characteristics of the driver output.

Noise bumps can occur in the following regions: below low, above low, below high, and above high. You can specify a steady-state resistance for each region.

This command can be used in the absence of library-specified I-V characteristics, or to override the library-specified characteristics by replacing them with a steady-state resistance value.

The **report_noise_calculation** command shows whether noise immunity information was taken from the library or annotated by this command.

EXAMPLES

This example specifies a steady state resistance of 2.0 for above-the-ground rail noise for the Z pin of the AN4 library cell in the lsi_10k library:

```
pt_shell> set_steady_state_resistance -above -low  2.0 lsi_10k/AN4/Z
```

SEE ALSO

remove_steady_state_resistance(2)
report_noise_calculation(2)

set_supply_voltage

Assigns a voltage to the supply nets used in rail analysis.

SYNTAX

```
status set_supply_voltage
   voltage_value
   -object_list supply_nets
```

Data Types

```
float voltage_value
list object_list
```

ARGUMENTS

voltage_value

Defines the voltage assigned to the supply nets.

object_list

Specifies a list of supply nets on which to set voltage.

DESCRIPTION

The **set_supply_voltage** command assigns ideal voltages to the supply nets during rail analysis. The command returns 1 if successful and 0 if unsuccessful.

This command is used only for rail analysis, Unlike the **set_voltage** command which is used for timing and power analysis, you can assign different voltages using the **set_supply_voltage** and **set_voltage** commands. This allows you to run timing, power and rail analysis with more flexibility.

The **set_supply_voltage** command is optional for rail analysis. If the **set_supply_voltage** command is not defined for a supply net, the tool derives the minimum voltage from all the cell instances connected to this supply net, and sets it as the ideal supply voltage for rail analysis. This represents the worst case for rail analysis. For example, there are three cell instances connected to the supply net VDD. One cell instance has voltage 1.0 and the other two cell instances have voltage 1.1 from timing and power analysis. The tool uses 1.0 for VDD in rail analysis, and reports multiple voltages (1.0 and 1.1) are connected to the same supply net with a warning.

PrimeRail uses the voltage value assigned by the **set_supply_voltage** command to calculate actual voltage

which is equal to this assigned voltage minus voltage drop (or plus voltage rise). This voltage value does not affect the voltage drop or rise result calculated during rail analysis.

As a comparison, PrimeRail uses the voltage value assigned by the **set_voltage** command to calculate static current as static power is divided by the voltage. Therefore, this voltage value affects the voltage drop or rise result calculated during rail analysis.

To check what supply voltage is used during rail analysis, use the **report_supply_voltage** command.

EXAMPLES

The following example sets the ideal voltage for the supply net VDD:

```
pr_shell> set_supply_voltage 1.2 -object_list VDD
```

The following example sets the ideal voltage for the supply net VDD and all the supply nets connected to VDD through power management cells. Then the the PG pin voltages are written to the vt_output file in the format of the **set_voltage** commands. The instance inverter_1 has voltage drop 0.15 volt on the VDD pin and is connected to the supply net VDD.

```
pr_shell> set_supply_voltage 1.2 -object_list [get_supply_nets -expand VDD]
pr_shell> report_rail_voltage -primetime vt_output
pr_shell> sh cat vt_output
set_voltage 1.05 -cell inverter_1 -pg_pin_name VDD
```

SEE ALSO

```
set_voltage(2)
get_supply_nets(2)
report_rail_voltage(2)
```

set_switched_net_state

Sets the state of the specified virtual supply nets.

SYNTAX

```
status set_switched_net_state
   virtual_supply_list
   -state state_name
   [ -start_voltage voltage_value ]
   [ -stop_voltage  voltage_value ]
   [ -threshold  threshold_value ]
```

Data Types

```
list virtual_supply_list
string state_name
float voltage_value
float threshold_value
```

ARGUMENTS

virtual_supply_list

Specifies a list of virtual supply nets for which to apply the specified state.

-state *state_name*

Specifies the state of the specified virtual supply nets. The valid state name can be *on*, *off*, or *ramp_up*.

-start_voltage *voltage_value*

Specifies the voltage value of the specified supply nets where ramp_up analysis starts.

-stop_voltage *voltage_value*

Specifies the voltage value of the specified supply nets where ramp_up analysis completes, i.e., the virtual supply nets become fully ramped up.

-threshold *threshold_value*

Specifies how the *stop_voltage* of a virtual supply net is calculated. If the -stop_voltage option is not specified. For a virtual power net, the *stop_voltage* is calculated by multiplying (1.0-threshold_value) to its operating voltage. For a virtual ground net, the *stop_voltage* is calculated by multiplying *threshold_value* to the related power supply voltage. When both the -stop_voltage and -threshold options are specified, the -threshold option is ignored.

The default *threshold_value* is 0.01, which is used when neither the -stop_voltage nor -threshold option is

specified.

DESCRIPTION

The command specifies the current state of virtual supply nets in the design. The valid state names are *on*, *off*, and *ramp_up*. By default, a virtual supply net is in **on** state.

When a virtual supply net is turned on or off using this command, the on-off state of the MTCMOS switch cells connected to the net is correspondingly changed. Therefore, a user can use this command to change the state of a virtual supply net without having to change the state of the MTCMOS switch cells individually.

The state of virtual supply nets is taken into consideration during power gated design analysis. See the **calculate_rail_voltage** command for details on power-gated analysis.

EXAMPLES

The following example changes the state of the virtual supply nets VDDV1 and VDDV2 to *off*. All MTCMOS switch cells connected to VDDV are turned off.

```
pr_shell> set_switched_net_state -state off {VDDV1 VDDV2}
1
```

The following example changes the state of the virtual supply net VDDV to *ramp_up*. It also specifies the start and stop voltages for the rush current analysis. When the state of a virtual supply net is set to *ramp_up*, rush current analysis is performed during power-gated analysis.

```
pr_shell> set_switched_net_state -state ramp_up \
          -start_voltage 0.1 -stop_voltage 1.0 VDDV
1
```

SEE ALSO

```
calculate_rail_voltage(2)
calculate_rush_currents(2)
report_annotated_mtcmos_switch_cell(2)
report_switched_net_state(2)
set_annotated_mtcmos_switch_cell(2)
```

set_switching_activity

Sets switching activity annotation on the selected nets, pins, ports, and cells of the current design.

SYNTAX

```
integer set_switching_activity
   [-state_condition state_condition]
   [-path_sources path_sources]
   [-toggle_rate toggle_rate]
   [-clock_derate derate_value]
   [-glitch_rate glitch_rate]
   [-static_probability static_probability]
   [-rise_ratio rise_ratio]
   [-period period_value]
   [-base_clock clock]
   [-type object_type_list]
   [-hierarchy]
   [-verbose]
   [-clock_domains clock_list]
   [object_list]
```

Data Types

state_condition	string
path_sources	list
toggle_rate	float
derate_value	float
glitch_rate	float
static_probability	float
rise_ratio	float
period_value	float
clock	string
object_type_list	list
clock_list	list
object_list	list

ARGUMENTS

-state_condition *state_condition*

Specifies the state condition when annotating state-dependent toggle rate and glitch rate on pins or state-dependent static probabilities on cells. State-dependent toggle rates and glitch rates can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal or leakage power characterization. The state condition should be enclosed between " ". To set switching activity for the default state condition, specify the argument of *-state_condition* option as *default*.

-path_sources *path_sources*

Specifies the path sources when annotating path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path sources specified with this argument must be the same as the path sources in the internal power characterization. When listing more than one pin in path sources, the pin names must be separated by a space and enclosed between " ". For example, if the pins in path sources are A and B, the argument for the *-path_sources* option must be "A B". Note: If you use the *-path_sources* option to specify the *path_sources value, then you must use the -state_condition option to specify the value of the path_sources.*

-static_probability *static_probability*

Specifies the value of the static_probability switching activity. The *static_probability* value represents the percentage of time the signal is at the logic state 1. For example, a *static_probability* of 0.25 indicates that the signal is in the logic state 1 for 25% of the time.

-toggle_rate *toggle_rate*

Specifies the value of the toggle rate switching activity. The rate is a floating point number that represents the number of 0->1 and 1->0 glitch free transitions, that the signal makes during a period of time. You can specify the period with the *-period* option. Alternatively, you can annotate a related clock with the *-base_clock* option, and the specified toggle rate is relative to the related clock period. If you specify the *-clock_domains* option, the related clock is chosen based on which clock domain the object belongs. If it belongs to multiple clock domains, the faster clock is the related clock. If no clock domain is found for the object, the fastest clock in the design is the related clock.

-clock_derate *derate_value*

The *-clock_derate* option is an alternative to the *-toggle_rate* option if the *-base_clock* option is specified. The annotated toggle rate for an object is a factor of the toggle rate of the related clock of the object. The related clock is chosen based on the clock domain to which the object belongs. If it belongs to multiple clock domains, the fastest one is selected. If the object does not belong to any clock domain, the related clock is set to the fastest clock in the design.

-glitch_rate *glitch_rate*

Specifies the value of the glitch rate switching activity. The value is a floating point number that represents the number of 0->1 and 1->0 glitch transitions, that the signal makes during a period of time. You can specify the period with the *-period* option. Alternatively, you can annotate a related clock using the *-base_clock* option, and the specified glitch rate is relative to the related clock period. Note: If either of the *-toggle_rate* or the *-glitch_rate* option is specified, the other is assumed to be zero.

For PrimeTime PX version D-2010.06 and earlier releases, the *toggle_rate* and *glitch_rate* options are known as *-toggle_count* and *-glitch_count* respectively. If you specify the older names, the tool generates a warning message.

-rise_ratio *rise_ratio*

Specifies the ratio of rise transitions to total transitions for the specified toggle rate and glitch rate when annotating pins that are characterized with both rise and fall internal power. The *rise_ratio* argument is a floating point number between 0.0 (all transitions are falling) and 1.0 (all transitions are rising). You need to specify a toggle rate and glitch rate to use this option. The default is 0.5.

-period *period_value*

Specifies the time period for which the number of transitions given in the *toggle_rate* and *glitch_rate* occur. The time units for this value are those specified by the main technology library. When you specify this option, the values of the specified toggle and glitch rates are divided by the given *period_value*. The resulting toggle rate and glitch rate are then annotated to the objects provided in the **set_switching_activity** command. If the *-period* option is not specified, a default *period_value* of 1.0 is assumed.

-base_clock *clock*

Specifies a clock by which the toggle and glitch rate values are referenced. When applied, the toggle and glitch rates are divided by the period of the specified clock. When the *-base_clock* value is set to "*", the toggle and glitch rates are divided by the period of the related clock.

-type *object_type_list*

Specifies a list of object types used for implicitly selecting the objects that are annotated with the specified switching activity. The *object_type_list* argument can be a list of one or more of the following object types:

- *registers* (sequential cell outputs)
- *three_states* (tristate cell outputs)
- *inputs* (input design ports / hierarchical instance pins)
- *outputs* (output design ports / hierarchical instance pins)
- *inout* (inout design ports / hierarchical instance pins)
- *ports* (design port / hierarchical instance pins)
- *nets* (nets)
- *clock_gating_cells* (clock-gating cell output)
- *black_boxes* (black box outputs)
- *non_clock_network* (nonclock network objects)
- *memory* (memory cell outputs)

When you specify the *-type* option, the tool annotates all the objects in the current instance (current design, if no current instance is set) that satisfy the selection criteria with the specified switching activity. If you specify the *registers* type, the tool annotates both the noninverting (Q) and inverting (QN) sequential cell outputs. The annotated toggle rate and glitch rate on the inverting outputs is the same as that on the noninverting outputs. The static probability on the inverting outputs is 1.0 - *value* where *value* is the specified static probability.

If you specify the *black_boxes* type, the tool annotates the black-box cell outputs. Generated clock nets are excluded from this group.

If you specify the *non_clock_network* type, the tool annotates the activity on the nonclock network cells of the design.

If you specify the *-clock_domains* option, the tool constrains the selected objects further to fall into the specified clock domains. However, the source pins of the clock are not included.

-hierarchy

Use with the *-type* option to specify that the objects in all the hierarchies in the current instance that satisfy the selection criteria. If not specified, the tool annotates only the top-level objects in the current instance that satisfy the selection criteria.

For PrimeTime PX version D-2010.06 and earlier releases, by default, the objects in all the hierarchies in the current instance are considered unless you specify the *-no_hierarchy* option; For PrimeTime PX version E-2010.12 and later, the *-no_hierarchy* option replaces the *-hierarchy* option. For the tool to consider all the hierarchies in the current instance, specify the *-hierarchy* option.

-verbose

Prints the detailed information, such as the design objects that are not annotated. By default, the tool suppresses detailed information.

For PrimeTime PX version D-2010.06 and earlier releases, verbose information was provided by default. A *-quiet* option was used to suppress the detailed information. In the E-2010.12 release, detailed information is suppressed by default; the *-quiet* option replaces the *-verbose* option. If you use the invalid *-quiet* option instead of the valid *-verbose* option, a warning message is generated.

-clock_domains *clock_list*

Filters the selection to include only the objects that belong to the specified clock domains and divides the toggle and glitch rate by the period of the related clock. If an object belongs to multiple clock domains, the fastest of the clocks, is selected as the related clock. Objects which belong to no clock domain, are automatically placed in the domain of the fastest design clock.

For PrimeTime PX version D-2010.06 and earlier releases, this option is called *-clocks*. For PrimeTime PX version E-2010.12 and later, the *-clocks* option is *-glitch_rate*. If you use the *-clocks* option, a warning message is issued.

object_list

Specifies a list of nets, pins, ports, or instances in the current design on which the *static_probability*, *toggle_rate*, and *glitch_rate* switching activity values are to be set. Use this option with the *-type* argument to specify that all the objects in the given list of instances that satisfy the selection criteria are annotated. This option can also be used with or without the *-no_hierarchy* option.

DESCRIPTION

Use this command to annotate design nets, ports, pins, and cells with the different kinds of switching activity. These include simple toggle rate, glitch rate, and static probability on nets, ports and pins; state and path dependent toggle rate and glitch rate on cell pins, and state dependent static probabilities on cells.

Toggle rates, glitch rates, and static probabilities are annotated by using the *-toggle_rate*, *-glitch_rate* and *-static_probability* options respectively. The toggle rate, glitch rate, and static probability can be made state-dependent by specifying the state condition with the *-state_condition* option. The toggle rate and glitch rate can be made path-dependent by specifying the path sources with the *-path_sources* option. You can specify a related clock using the *-base_clock* option. The annotated toggle rate and glitch rate are divided by 1 time unit defined in the main library unless the -period, *-base_clock* or *-clock_domains* options are applied. When you specify the *-period* option, the toggle and glitch rates are divided by the period value. When you specify *-base_clock* option, the toggle and glitch rates are divided by the clock period of the specified base clock. When you specify the *-clock_domains* option, the glitch and toggle rates are divided by the period of the related clock for each domain.

The design objects that are annotated with switching activity can be specified explicitly as a list of objects. The objects can also be specified implicitly by using the *-type*, and *object_list* options together.

For statistics on the switching activity annotation on the current design, use the **report_switching_activity** command.

EXAMPLES

In the following example, tool annotates a simple toggle rate value of 33 / 1320 = 0.025, glitch rate value of 10 / 1320 = 0.0075 and a static probability value of 0.015 to all design input ports.

```
pt_shell> set_switching_activity -toggle_rate 33 -glitch_rate 10 \
          -period 1320 -static_probability 0.015 [all_inputs]
```

The following example annotates the same activity to all input ports, relative to a clock with a period of 10.

pt_shell> **create_clock CLK -period 10** pt_shell> **set_switching_activity -toggle_rate .25 -glitch_rate .05 \ -base_clock CLK -static_probability .015 -type inputs**

The toggle rate value of 0.25 and glitch rate value of 0.05 are divided by clock period (10). The resulting toggle rate value of 0.025 and glitch rate value of 0.005 are annotated on the design objects.

In the following example, the tool annotates the state_condition dependent static probabilities on the cell or1:

```
pt_shell> set_switching_activity -static_probability 0.10 \
           -state_condition "A & B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.25 \
           -state_condition "A & ! B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.25 \
           -state_condition "! A & B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.30 \
           -state_condition "! A & ! B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.10 \
           -state_condition "default" [get_cell or1]
```

In the following example, the tool annotates simple and path dependent toggle rates and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> set_switching_activity -toggle_rate 0.022 -glitch_rate 0.001 \
           [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.020 -glitch_rate 0.001 \
           -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.002 -glitch_rate 0.0001 \
           -path_sources "B" -state_condition "A" [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.002 -glitch_rate 0.0001 \
           -path_sources "A B" -state_condition "default" [get_pin xor1/Y]
```

In the following example, the command annotates a toggle rate which is a fraction of the related clock on all the clock-gating cells.

```
pt_shell> set_switching_activity -clock_derate 0.1 \
           -clock_domains [all_clocks] \
           -type clock_gating_cells
```

The clock-gating cell outputs are annotated with a toggle rate which is 0.1 of the input clock toggle rate. If the input clock period is 10ns, its toggle rate is 2 toggle/10ns = 0.2 and the toggle rate on the clock gating output is 0.1 * 0.2 = 0.02.

SEE ALSO

```
create_clock(2)
get_switching_activity(2)
read_saif(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
```

set_tap_package_connection

Specifies the connection among a package SPICE model port and the taps.

SYNTAX

```
status set_tap_package_connection
   taps
   -port_name port_name
   [-no_mapping]
```

Data Types

```
collection taps
string port_name
```

ARGUMENTS

taps

Specifies a collection of taps which connect to the specified package SPICE model port.

-port_name *port_name*

Specifies a port name of the package SPICE model.

-no_mapping

Ignores actual connection among the specified port and taps and honors only the *package_port_name* attribute of the specified taps that is set to the specified port. You can specify the actual connection with the the **set_tap_package_model** command. The option is required if the connection is defined before package file is specified.

DESCRIPTION

The **set_tap_package_connection** command specifies the mapping among a package SPICE model port and its connected taps. You specify the name of the package model port with the *-port_name* option and the taps with the *taps* argument. The specified taps can be a collection of taps obtained by using the **get_taps** command.

EXAMPLE

The following example connects the PKG_VDD port of the package model to all the taps of the VDD net on the die (that is, the core).

```
pr_shell> set_tap_package_connection -port_name PKG_VDD [get_taps -of VDD]
```

The following example connects the PKG_VDD (PKG_VSS) port of the package model to all taps of the VDD (VSS) net on the die. -no_mapping is used here to avoid checking for actual connection among the package port and the taps when **set_tap_package_model** command is run after **set_tap_package_connection**.

```
pr_shell> set_tap_package_connection -port_name PKG_VDD -no_mapping [get_taps -of VDD]
pr_shell> set_tap_package_connection -port_name PKG_VSS -no_mapping [get_taps -of VSS]
pr_shell> set_tap_package_model -spice package_model.sp
```

SEE ALSO

create_taps(2)
get_taps(2)
report_taps(2)
set_tap_package_model(2)

set_tap_package_model

Creates package models by setting the parasitic resistance, capacitance, and inductance values to the tap points or the package SPICE models.

SYNTAX

```
status set_tap_package_model
   [-R value]
   [-L value]
   [-C value]
   [-type type_name]
   tap_list
   [-spice spice_file_name]
   [-spice_include_path spice_include_path]
   [-spice_top spice_top_name]
```

Data Types

```
float value
string type_name
list taps
string spice_file_name
list spice_include_path
string spice_top_name
```

ARGUMENTS

-R *value*

Assigns the specified resistance value in the library units to the taps.

-L *value*

Assigns the specified inductance value in the library units to the taps.

-C *value*

Assigns the specified capacitance value in library units to the taps.

-type *type_name*

Specifies the parasitic component model when two or more parasitic elements, including resistance, induction or capacitance, are applied to the specified taps. The valid *typename* values are *RCL*, *CRL*, or *LCR*. Each parasitic model type is determined based on the sequence of the model components when viewed from the tap. For instance, the RCL type indicates that the tap parasitics are modeled as a resistor, a capacitor and an inductor in series, with the resistor connected to the tap and the inductor connected to an external ideal voltage source. The default is *CRL*.

tap_list

Specifies a list of taps on which parasitic values are to be applied. Each list element can be a tap collection or a tap name pattern.

-spice ***spice_file_name***

Specifies the SPICE package model file in which each packages is described as a SUBCKT device with power and ground net tap points as ports. The SPICE models are compatible with HSPICE.

.IP "-spice_include_path *spice_include_path* Specifies the search path to the combined SPICE package models that are included with the .INCLUDE statement. This option is valid only when the -spice option is specified.

-spice_top ***spice_top_name***

Specifies the top sub-circuit (.SUBCKT) of the package model if the SPICE file has multiple top sub-circuits. This option is valid only when the -spice option is specified.

DESCRIPTION

The **set_tap_package_model** command provides a number of built-in electrical models that are effective in modeling many package configurations. You can select these built-in models by using the -type *type_name* option with the -R,-L, and -C options to specify the parasitic values of the various model components. The parasitic values are in library units. To display the library unit, use the **report_units** command.

Use the -type option to specify an electrical model for different component configurations, depending on the sequence of the resistance, capacitance, and inductance values, and the connection between an external ideal voltage source and a tap point. The diagrams below show the component configurations for the supported model types: CRL, RCL, and LCR.

```
     +---+           +-----------+     +-----------+     +---+ external
CRL: |TAP|-----+-----|Resistor(R)|-----|Inductor(L)|-----| V | ideal
     +---+     |     +-----------+     +-----------+     +---+ voltage
         -------------
          Capacitor(C)
         -------------
               |
              GND

     +---+     +-----------+         +-----------+     +---+ external
RCL: |TAP|-----|Resistor(R)|----+----|Inductor(L)|-----| V | ideal
     +---+     +-----------+    |    +-----------+     +---+ voltage
                            -------------
                             Capacitor(C)
                            -------------
                                  |
                                 GND

     +---+    +-----------+         +-----------+    +---+ external
LCR: |TAP|----|Inductor(L)|----+----|Resistor(R)|----| V | ideal
     +---+    +-----------+    |    +-----------+    +---+ voltage
                           -------------
                            Capacitor(C)
                           -------------
                                 |
                                GND
```

The -R,-L, and-C options are used to define the resistance, inductance, and capacitance values for the model components. When an -R,-L, and -C option is not specified, by default the value is set to 0, meaning the corresponding model component is not used.

When modeling a single electrical component between the tap and the external ideal voltage, specify either of -the R,-L, and -C options. In this case, specifying the -type option is not required.

The **set_tap_package_model** command also supports user-defined SPICE package model that is compatible with HSPICE. The following are the supported models:

```
Passive element models  R, L, and C
Independent source models I and V
Dependent source models E, F, G, and H with linear dependency
```

To create a SPICE package model, users have to prepare a SPICE file with a top subcircuit (.SUBCKT) as an interface to connect the taps of DUA. To connect the package to the DUA physically, run the **create_taps** command to create taps and connect them to package ports by **set_tap_package_connection**.

For example, assume the top subcircuit of package model has 2 ports: PKG_VDD1 and PKG_VDD2 ports that connect to the differnet taps of VDD net of the DUA. In this case, the user needs to run **create_taps** command to create TAP1 and TAP2 for the PKG_VDD1 and PKG_VDD2 ports. Then run **set_tap_package_connection** to connect TAP1 to PKG_VDD1 and TAP2 to PKG_VDD2.

For a hierarchical SPICE circuit, PrimeRail uses the root subcircuit in the hierarchy tree as the top circuit. To assign another subcircuit or multiple top subcircuits are available, use the -spice_top option to specify a subcircuit as the top subcircuit.

When the combined SPICE files are included with the .INCLUDE statement, use the -spice_include_path option to specify the search path.

Please note when a SPICE package model is set in the design, PrimeRail ignores the taps that are not related to the package model during rail analysis.

EXAMPLE

The following example sets a resistance value of 5 units (as defined by the technology library units) on a tap point:

```
pr_shell> set_tap_package_model -R 5 TAP_23
```

The following example sets a resistance value of 0.5 units, and an inductance value of 5 units (as defined by the technology library units) on a tap point:

```
pr_shell> set_tap_package_model -type CRL -R 0.5 -L 5 TAP_24
```

The following example shows how to set a user-defined package SPICE model file, named simple.sp, on the VDD net.

1. First prepare a SPICE package model. The following is the content of the simple.sp SPICE file in which a subcircuit package model is described with two ports connecting to the VDD net.

```
pr_shell> exec cat simple.sp

\[char46]SUBCKT package pkg_vdd1 pkg_vdd2

R1_1 pkg_vdd1 1 1.0
L1_1 1 v 1.0e-9
C1_1 pkg_vdd1 0 1.0e-9

R2_1 pkg_vdd2 2 1.0
L2_1 2 v 1.0e-9
```

```
C2_1 pkg_vdd2 0 1.0e-9

RB 1 2  1.5
LB  1 2  1.0e-9

Vdd v 0 1.5

\[char46]ENDS
```

2. Run the following command to set the package model:

```
pr_shell> set_tap_package_model -spice simple.sp -no_mapping
```

3. Run the following command to create taps for the package ports:

```
pr_shell> create_taps -name TAP1 -layer 19 -point {540 597} -supply_net VDD
pr_shell> create_taps -name TAP2 -layer 19 -point {25 597} -supply_net VDD

And connect the tap with the package portby set_tap_package_connection.
Please be noted the SPICE port name must be capital.

pr_shell> set_tap_package_connection -port_name PKG_VDD1 [get_taps TAP1]
pr_shell> set_tap_package_connection -port_name PKG_VDD2 [get_taps TAP2]

Or you can create taps for the package ports by defining the information in a
user-defined tap file, and then import the file with the create_taps command.

For example, you define the information in a user-defined tap file with the following
content:

VDD 19 540 597 PKG_PORT PKG_VDD1
VDD 19 25  597 PKG_PORT PKG_VDD2

Save the file in the name user_define_tap. Run the following command to create taps
with this file:


pr_shell> create_taps -import user_define_tap
```

The following command reports the current package model configuration and the connected package port for rail analysis.

```
pr_shell> report_taps -package
****************************************
Report : rail analysis tap package
Design : chip
Version: H-2012.12-LCA-DEV
Date   : Wed Nov 14 17:46:58 2012
****************************************

Package spice subcircuit: PACKAGE
Package ports               : PKG_VDD1 PKG_VDD2
(*) floating package ports

Name      Type           Net            Port
--------------------------------------------------------------------------------
TAP1      absolute_coord VDD            PKG_VDD1
TAP2      absolute_coord VDD            PKG_VDD2
```

SEE ALSO

```
calculate_rail_voltage(2)
create_taps(2)
set_tap_package_connection(2)
report_taps(2)
report_units(2)
```

set_temperature

Applies an operating temperature on a list of cell objects.

SYNTAX

```
int set_temperature
   max_case_temperature
   [-min min_case_temperature]
   -object_list list_of_cells
```

Data Types

```
max_case_temperature      float
min_case_temperature      float
list_of_cells             list
```

ARGUMENTS

max_case_temperature

Specifies the operating temperature for the maximum (worst) case.

-min *min_case_temperature*

Specifies the operating temperature for the minimum (best) case.

-object_list *list_of_cells*

Specifies the list of cells that have the operating temperatures as specified in this command.

DESCRIPTION

Defines the operating temperature on the cells so that these cells of the design are timed and optimized at the specified temperature. If you do not specify any operating temperature for the cells, they continue to be timed and optimized based on the available operating condition settings.

If *temperature_ranges* are specified for a cell, the operating temperatures of the cell must fall within one of the ranges specified in the *temperature_ranges*. Furthermore, *min_case_temperature* must fall within the same range as the *max_case_temperature*.

The operating temperature of a cell, once defined, can only be overridden with the **set_temperature** command used on a cell. It can also be cleared using the **reset_design** command.

EXAMPLES

The following example shows a typical context for using the **set_temperature** command, and has the max_case_temperature set at 125 on cell I1:

```
pt_shell> set_temperature 125 \
           -object_list [get_cells I1]
1
```

The following example sets a max_case_temperature of 125 and min_case_temperature of 25 on cell I2.

```
pt_shell> set_temperature 125 \
         -min 25 \
         -object_list [get_cells I2]
1
```

SEE ALSO

set_voltage(2)

set_timing_derate

Adjusts the calculated cell and net delays by a specified factor to model the effects of on-chip variation.

SYNTAX

```
integer set_timing_derate
   -early | -late
   [-rise] [-fall]
   [-data] [-clock]
   [-cell_delay] [-net_delay] [-cell_check]
   [-static] [-dynamic]
   [-scalar] [-variation]
   [-aocvm_guardband] [-pocvm_guardband] [-pocvm_coefficient_scale_factor]
   [-increment]
   derate_value
   [object_list]
```

Data Types

derate_value	float
object_list	list

ARGUMENTS

-early

Applies the derating to the early delays (shortest paths), such as cell and net delays in the data path of a hold check.

-late

Applies the derating to the late delays (longest paths), such as cell and net delays in the data path of a setup check.

The command must specify either **-early** or **-late**. To set both early and late derating, use two separate **set_timing_derate** commands.

-rise

Applies the derating only to cell and net delays that end with a rising transition. By default, both rising and falling delays are derated. To specify different derating values for rising and falling delays, use two separate **set_timing_derate** commands.

-fall

Applies the derating only to cell and net delays that end with a falling transition.

-data

Applies the derating only to delays in data paths. By default, delays are derated in both data paths and clock paths. To specify different derating values for data paths and clock paths, use two separate **set_timing_derate** commands.

-clock

Applies the derating only to delays in clock paths.

-cell_delay

Applies the derating only to cell delays. By default, derating applies to both cell and net delays, but not to cell timing check constraints.

-net_delay

Applies the derating only to net delays.

-cell_check

Applies the derating to cell timing check constraints instead of delays. Derating set with the **-early** option applies to hold and removal timing checks. Derating set with the **-late** option applies to setup and recovery timing checks. The **-cell_check** option cannot be used with the **-clock**, **-data**, **-cell_delay**, **-net_delay**, or **-aocvm_guardband** options.

-static

Applies the derating to only the static component of net delay, which is the net delay without considering crosstalk. This option works only with the **-net_delay** option. By default, both the static and dynamic delay components are derated.

-dynamic

Applies the derating to only the dynamic component of net delay, which is the change in delay (delta delay) caused by crosstalk. This option works only with the **-net_delay** option and cannot be used with the **-aocvm_guardband** or **-pocvm_guardband** option.

-scalar

Applies the derating only to deterministic delays. This option should be used only in the PrimeTime VX tool.

-variation

Applies the derating only to statistical delays. This option should be used only in the PrimeTime VX tool.

-aocvm_guardband

Applies the derating on top of advanced on-chip variation (AOCV) derating. The total derating applied to a timing arc is the product of the AOCV derating factor and the AOCV guardband factor. This option has an effect only when the **timing_aocvm_enable_analysis** variable is set to **true**, and only affects the timing arcs derated by AOCV analysis. The **-aocvm_guardband** option cannot be used with the **-dynamic** or **-cell_check** option.

-pocvm_guardband

Applies the derating on top of parametric on-chip variation (POCV) analysis. The specified derating factor applies to both the mean and standard deviation of the statistical delay distribution. This option has an effect only when the **timing_pocvm_enable_analysis** variable is set to **true**, and only affects the timing arcs analyzed by POCV analysis. The **-pocvm_guardband** option cannot be used with the **-dynamic** option.

-pocvm_coefficient_scale_factor

Applies the derating to the variation coefficient C of the statistical delay distribution in parametric on-chip

variation (POCV) analysis. This effectively scales the sigma without affecting the mean. This option has an effect only when the **timing_pocvm_enable_analysis** variable is set to **true**, and only affects the sigma of timing arcs analyzed by POCV analysis. The **-pocvm_coefficieint_scale_factor** option cannot be used with the **-dynamic** option.

To report the settings for AOCV guardband, POCV guardband, or POCV coefficient scaling, use the **report_timing_derate** command with the **-aocvm_guardband**, **-pocvm_guardband**, or **-pocvm_coefficient_scale_factor** option.

-increment

Applies the derating on top of a previous derating setting. The specified value is added to (not multiplied by) the existing derating value previously set, or added to 1.0 if no derating value was previously set. Use this option to selectively adjust the existing derating settings for selected objects (cells, library cells, or nets) or conditions (**-cell_delay**, **-net_delay**, **-rise**, **-fall**, and so on). The **-increment** option cannot be used with AOCV/POCV options (**-aocvm_guardband**, **-pocvm_guardband**, and **-pocvm_coefficient_scale_factor**).

To report the incremental derating settings, use the **report_timing_derate** command with the **-increment** option. To see the composite (combined) derating values applied to individual cell and net delays, use the **report_timing** command with the **-derate** option.

derate_value

Specifies the derating value. This value is multiplied by the calculated cell and net delays (or timing constraint values if the **-cell_check** option is used), or added to the existing derating factor if the **-increment** option is used.

object_list

Specifies a list or collection of cells, library cells, or nets. The derating factor applies only to the specified objects. By default, the derating factor applies to all cell and net delays in the current design.

DESCRIPTION

The **set_timing_derate** command modifies calculated delays, allowing you to model the effects of on-chip variation. The effect is to multiply the calculated delays by a factor that you specify, which changes the delay and slack values reported by the **report_timing** command and other reporting commands.

The **set_timing_derate** command specifies the early or late delay adjustment factor and optionally the scope of the design affected by derating. For example,

```
pt_shell> set_timing_derate -early 0.9
pt_shell> set_timing_derate -late 1.2
```

The first command decreases all early (shortest-path) cell and net delays by 10 percent, such as those in the data path of a hold check. The second command increases late (longest-path) cell and net delays by 20 percent, such as those in the data path of a setup check. These two adjustments result in a more conservative analysis.

Using the **set_timing_derate** command implicitly changes the tool to on-chip variation mode, if not already in that mode, like using the following command:

pt_shell> **set_operating_conditions -analysis_type on_chip_variation**

In on-chip variation mode, the tool applies worst-case delay adjustments, both early and late, at the same time. For example, for a setup check, it applies late derating on the launch clock path and data path, and early derating on the capture clock path. For a hold check, it does the opposite.

In the **set_timing_derate** command, you must use either **-early** or **-late** to specify either shortest-path or longest-path delays for applying the derating. To set both early and late derating, use two separate **set_timing_derate** commands.

You must also specify the derating factor, a floating-point number. To get a more conservative analysis that considers the effects of on-chip variation, use a derating factor less than 1.0 for early derating or greater than 1.0 for late derating.

To report derating that has been set, use the **report_timing_derate** command.

To report the derating factor used for each incremental delay in a timing report, use the **-derate** option in the **report_timing** command:

```
pt_shell> report_timing -derate
 ...
 Path Type: max

 Point                                          Derate   Incr      Path
 ----------------------------------------------------------------------
 clock PCI_CLK (rise edge)                                0.000     0.000
 clock network delay (propagated)                         1.802     1.802
 I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)             0.000     1.802 r
 I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)    1.200     0.485 &   2.287 r
 U7/I (inv0d1)                                  1.200     0.010 &   2.297 r
 U7/ZN (inv0d1)                                 1.100     0.063 &   2.360 f
 U62/I (inv0d1)                                 1.100     0.005 &   2.365 f
 U62/ZN (inv0d1)                                1.200     0.046 &   2.410 r
 U63/A2 (or02d7)                                1.200     0.005 &   2.415 r
 U63/Z (or02d7)                                 1.200     0.318 &   2.733 r
 ...
```

To cancel all derating, use the **reset_timing_derate** command.

Derating Options

By default, the **set_timing_derate** command applies the specified early or late derating factor to all early or late timing paths in the whole design. You can optionally modify the scope of the command in several ways:

By default, the **set_timing_derate** command applies the specified early (or late) derating factor to all cell and net delays in early (or late) timing path segments in the design. You can optionally modify the scope of the command in several ways:

- To derate only clock paths or only data paths, use the **-clock** or **-data** option.
- To derate only net delays or only cell delays, use the **-net_delay** or **-cell_delay** option.
- To derate only rising-edge delays, use the **-rise** option. Then the derating applies only to cell and net delays that end with a rising transition. Similarly, to derate only falling-edge delays, use the **-fall** option.
- To derate only certain nets, cell instances, or library cells, enter an object list in the command. You can use an embedded **get_nets**, **get_cells**, or **get_lib_cells** command to generate the list.
- To derate only the static or dynamic component of net delay, use the -**net_delay** option together with the **-static** or **-dynamic** option. By default, both the static and dynamic components are derated. The static component is the net delay without considering crosstalk. The dynamic component is the change in net delay (delta delay) caused by crosstalk.
- To derate cell timing check constraints instead of delays, use the **-cell_check** option. Derating set with the **-early** option applies to hold and removal timing checks. Derating set with the **-late** option applies to setup and recovery timing checks. A **derate_value** greater than 1.0 increases the value of the timing constraint, resulting in a more conservative check. By default, cell timing checks are not derated.

- To apply derating to advanced on-chip variation (AOCV) analysis or parametric on-chip variation (POCV) analysis, use the **-aocvm_guardband**, **-pocvm_guardband**, or **-pocvm_coefficient_scale_factor** option. By default, the **set_timing_derate** command affects ordinary cell and net delays, not AOCV or POCV analysis.

To report the derating options that have been set, use the **report_timing_derate** command. For example,

```
pt_shell> report_timing_derate -significant_digits 2
 ...
                          ----- Clock -----             ----- Data -----
                         Rise          Fall            Rise          Fall
                       Early  Late    Early  Late    Early  Late    Early  Late
-------------------------------------------------------------------------------
design: MYDESIGN
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Net delay dynamic   0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Cell delay          0.90   1.20    0.90   1.10    0.90   1.20    0.90   1.10
  Cell check           --     --      --     --      --     --      --     --

net: net198
  Net delay static    0.90   1.30    0.90   1.30    0.90   1.30    0.90   1.30
  Net delay dynamic   0.90   1.30    0.90   1.30    0.90   1.30    0.90   1.30
```

If you set derating on a net or leaf-level cell, the command implicitly matches the derating type to the object type: net derating for a net, or cell derating for a leaf-level cell. Derating set on a net applies to the whole net, even as it crosses into a lower level of hierarchy.

To set derating on a hierarchical cell, you must explicitly specify the type of derating by using the **-net_delay** or **-cell_delay** option (or both), or the **-cell_check** option.

To enable derating of minimum pulse width and minimum period constraints, set the **timing_use_constraint_derates_for_pulse_checks** variable to true. In that case, the tool applies the derating factors set with the **-cell_check** and **-late** options of the **set_timing_derate** command.

Conflicting Derating Settings

A new **set_timing_derate** command overrides any derating value set previously on the same scope of the design. In case of conflicting settings that overlap in scope, the command with the narrower scope has priority. In the following script example, the earlier commands are more specific in scope than the later ones, so the earlier ones have priority.

```
# Derate a collection of cell instances
set_timing_derate -late -cell_delay 1.4 [get_cells U2*]
#
# Derate a collection of library cells
set_timing_derate -late -cell_delay 1.3 [get_lib_cells libAZ/ND*]
#
# Derate all cell delays
set_timing_derate -late -cell_delay 1.2
#
# Derate the design (all cell and net delays)
set_timing_derate -late -1.1
```

Derating Negative Delays

Delays can be negative in some unusual situations. For example, if a cell's input transition is slow and its output transition is fast, and if the input-to-output delay is very short, the output signal can reach the 50 percent trip point before the input signal, resulting in a negative delay for the cell. A similar situation can occur for a change in net delay caused by crosstalk.

In general, delays are adjusted according to the following formula:

```
delay_new = old_delay + ( (derating_factor - 1.0) * abs(old_delay) )
```

When the delay is a positive value (the usual case), this equation is reduced to:

```
delay_new =  old_delay * derating_factor
```

For negative delay, the delay adjustment equation is reduced to:

```
delay_new =  old_delay * ( 2.0 - derating_factor )
```

When both the static and dynamic components of a net delay are derated, the two components are derated separately first, and then combined, so that a negative delta delay is properly derated before it is combined with a positive static delay.

EXAMPLES

The following commands set an early (shortest path) derating factor of 0.70 and a late (longest path) derating factor of 1.05 globally on all cell and net delays the design:

```
pt_shell> set_timing_derate -early 0.70
pt_shell> set_timing_derate -late 1.05
```

The following command sets an early derating factor of 0.75 on a specific net, n123, overriding the global derating factor for net delays:

```
pt_shell> set_timing_derate -net_delay -early 0.75 [get_nets n123]
```

The following command sets an early derating factor of 0.80 for the cell delay of cell instance U1, overriding the global derating factor for cell delays:

```
pt_shell> set_timing_derate -early 0.80 -cell_delay [get_cells U1]
```

The first command below sets a late derating factor of 1.10 on all instances of library cell IV in the library MY_LIB. The next command overrides that setting for one particular instance of that cell, inv82:

```
pt_shell> set_timing_derate -late 1.10 [get_lib_cells MY_LIB/IV]
pt_shell> set_timing_derate -late 1.05 [get_cells inv82]
```

The following command sets a late derating factor on all cells and nets in the hierarchical cell top/H1, including its lower-level hierarchical cells:

```
pt_shell> set_timing_derate -cell_delay -net_delay -late 1.05 [get_cells top/H1]
```

The following script sets global late and early derating factors, and then overrides those settings for particular cell instances:

```
set_timing_derate -late 1.09
set_timing_derate -early 0.91
set_timing_derate -late 1.11 [get_cells u1/u252]
set_timing_derate -increment -late 0.04 [get_cells u1/u252]
set_timing_derate -increment -early -0.03 [get_cells u1/u252]
```

These commands set a late derating factor of 1.15 (1.11 + 0.04) and an early derating factor of 0.88 (0.91 + (-0.03)) for the cell delay of cell u1/u252.

SEE ALSO

```
report_design(2)
report_timing(2)
report_timing_derate(2)
reset_timing_derate(2)
set_operating_conditions(2)
si_use_driving_cell_derate_for_delta_delay(3)
timing_aocvm_enable_analysis(3)
timing_enable_cumulative_incremental_derate(3)
timing_pocvm_enable_analysis(3)
```

set_units

Checks the specified units with the main library units.

SYNTAX

```
status set_units
   [-time time_in_sec]
   [-capacitance capacitance_in_farad]
   [-current current_in_ampere]
   [-voltage voltage_in_volt]
   [-resistance resistance_in_ohm]
   [-power power_in_watt]
```

Data Types

```
time_in_sec                 float
capacitance_in_farad        float
current_in_ampere           float
voltage_in_volt             float
resistance_in_ohm           float
power_in_watt               float
```

ARGUMENTS

-time *time_in_sec*

Checks the time unit with the time unit of the main library.

-capacitance *capacitance_in_farad*

Checks the capacitance unit with the capacitance unit of the main library.

-current *current_in_ampere*

Checks the current unit with the current unit of the main library.

-voltage *voltage_in_volt*

Checks the voltage with the voltage unit of the main library.

-resistance *resistance_in_ohm*

Checks the resistance unit with the resistance unit of the main library.

-power *power_in_watt*

Checks the power unit with the leakage power unit of the main library. The power units can only be checked if power analysis mode is enabled.

DESCRIPTION

The **set_units** command checks the specified units with the main library units. The command issues a warning if the specified units do not match the main library units. You can use this command only after linking the design.

The main library units are the design units. The **set_units** command does not set the design units, but it prevents the inadvertent use of the wrong units from different SDC files.

The power units can only be checked if power analysis mode is enabled. This can be done by setting the **power_enable_analysis** variable to **true** and running power related commands, such as **update_power**, **read_vcd**, **read_saif**, and **set_switching_activity**.

To report all units, run the **report_units** command. The **read_sdc** command supports the **set_units** command. The **write_sdc** and **write_script** commands write the **set_units** command as the first command at their output.

The **set_units** command does not affect the output of reporting commands. Reports continue to use the units defined in the main library.

EXAMPLES

The following example checks the if the main library time unit is 1ns, capacitance unit is 1pF, current unit is 1mA and voltage unit is 1V.

```
pt_shell> set_units -time ns -capacitance pF -current mA -voltage V
```

The following example checks if the main library capacitance unit is 0.5pF.

```
pt_shell> set_units -capacitance 0.5pF
```

SEE ALSO

```
read_sdc(2)
report_units(2)
write_script(2)
write_sdc(2)
```

set_user_attribute

Sets a user attribute to a specified value on an object.

SYNTAX

```
string set_user_attribute
   [-class class_name]
   [-quiet]
   object_spec
   attr_name
   value
```

Data Types

```
class_name        string
object_spec       list
attr_name         string
value             string
```

ARGUMENTS

-class *class_name*

If the *object_spec* option is a name, this is its class. Allowable values are design, port, cell, pin, net, lib, lib_cell, or lib_pin.

-quiet

Suppresses all report messages.

object_spec

Objects on which to set the attribute. Each element in the list is a collection or a pattern which is combined with the *class_name* option to find the objects.

attr_name

Shows the name of the attribute.

value

Shows the value of the attribute.

DESCRIPTION

The **set_user_attribute** command sets attributes on objects. These attributes are defined with the **define_user_attribute** command.

You cannot set application attributes with this command. Each settable application attribute has a command dedicated to it. For example, the *fanout_load* attribute is set with the **set_fanout_load** command. However, you can still retrieve the values of any application or user-defined attribute using the **get_attribute** command.

EXAMPLES

This example defines an attribute 'X' for cells, then sets the value on all cells in this level of the hierarchy.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
```

SEE ALSO

```
collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
remove_user_attribute(2)
report_attribute(2)
```

set_user_units

This command sets the units that are used for user input and output.

SYNTAX

```
string set_user_units
   -type unit_type
   -value unit_value

string unit_type
string unit_value
```

ARGUMENTS

-type *unit_type*

Specifies the type of unit to set. This is one of "capacitance", "current", "time" or "voltage".

-value *unit_value*

Specifies the unit value. For time, equivalent examples would include "1ps", "1.0ps", "0.001ns".

DESCRIPTION

There are unit values for time, capacitance, voltage and current. This command sets the units that are used for invoking commands **set_rail_current** and **set_intrinsic_rc** before **open_mw_cel** or **open_block**. When commands "**set_rail_current** and **set_intrinsic_rc** are invoked after **open_mw_cel** or **open_block**, the units are derived from the first logic .db from the link path, which may be different from the units in the command **set_user_units**.

EXAMPLES

The following example sets units to 10ps, 1pF, 1V and 1mA.

```
prompt> set_user_units -type time -value 10ps
prompt> set_user_units -type capacitance -value 1pF
prompt> set_user_units -type voltage -value 1V
prompt> set_user_units -type current -value 1mA
```

The following example sets time units to 1ps.

```
prompt> set_user_units -type time -value 1ps
```

SEE ALSO

set_vectorfree_rail_analysis_options

Sets the options for running the dynamic vectorfree rail analysis.

SYNTAX

```
    int set_vectorfree_rail_analysis_options
    [-simulation_time simulation time in ns]
    [-number_of_cycles number of clock cycles]

float   simulation time in ns
int     number of clock cycles
```

ARGUMENTS

-simulation_time *simulation time in ns*

Specifies the total time in nanoseconds for which the design needs to be simulated in the dynamic vectorfree rail analysis.

-number_of_cycles *number of clock cycles*

Specifies the number of dominant clock cycles for which the design needs to be simulated in dynamic vectorfree rail analysis.

DESCRIPTION

This command allows the user to set the duration for which the simulation needs to be done in the dynamic vectorfree rail analysis. The duration can be specified either in terms of the time in nanoseconds or the number of the dominant clock cycle of the design.

EXAMPLES

The following example performs dynamic vectorfree rail analysis for 10 nanoseconds.

```
pr_shell> set_vectorfree_rail_analysis_options -simulation_time 10
pr_shell> update_currents -dynamic_vectorfree
```

The following example performs dynamic vectorfree rail analysis for 2 dominant clock cycles.

```
pr_shell> set_vectorfree_rail_analysis_options -number_of_cycles 2
pr_shell> update_currents -dynamic_vectorfree
```

SEE ALSO

update_currents(2)

set_voltage

Applies a voltage on a list of supply nets or PG pins.

SYNTAX

```
status set_voltage
   [-min min_case_value]
   [-min_dynamic dynamic_min_case_value]
   [-dynamic dynamic_max_case_vlaue]
   [-object_list list_of_supply_nets]
   [-cell cell]
   [-pg_pin_name pg_pin_name]
   max_case_value
```

Data Types

```
float max_case_voltage
float min_case_voltage
float dynamic_max_case_voltage
float dynamic_min_case_voltage
list list_of_supply_nets
list cell
list pg_pin
list object_list
list supply_net
```

ARGUMENTS

-max *max_case_voltage*

Specifies the voltage for the maximum delay (slowest) case. This is typically a smaller voltage value.

-min *min_case_voltage*

Specifies the voltage for the minimum delay (fastest) case. This is typically a larger voltage value.

-dynamic *dynamic_max_case_value*

-min_dynamic *dynamic_min_case_value*

Specifies a portion of the voltage change that varies with the frequency significantly higher than several clock periods. The dynamic portion is considered during capacitive coupling analysis and various pessimism removal optimization.

-object_list *list_of_power_nets*

Specifies a list of supply nets for which the specified voltage is to be applied. If a supply net is connected to the virtual supply nets through MTCMOS switch cells, the applied voltage value is automatically expanded to those

virtual nets.

-cell ***cell_name***

Specifies the name of the cell.

-pg_pin_name ***pg_pin***

Specifies the name of the power or ground pin for voltage drop annotation.

DESCRIPTION

The command defines the voltage on the supply nets or PG pins so that the parts of the design powered by these supply nets are timed, optimized, and analyzed using the specified voltage.

Note that the voltage value defined in this command also affects the power and current values that are calculated by the **update_currents** command.

EXAMPLES

The following example sets the voltage for the supply net VDD.

```
pr_shell> set_voltage 0.88 -min 0.88 -object_list VDD
```

Assuming VDD is connected to the virtual nets VVDD1 and VVDD2 through MTCMOS switch cells, the following example sets the voltage for VDD, VVDD1, and VVDD2 without specifying VDD1 and VDD2 explicitly.

```
pr_shell> set_voltage 1.0 -min 1.0 -object_list VDD
```

The following command sets the voltage value on a PG pin VDD of the cell INV1.

```
pr_shell> set_voltage 0.44 -min 0.45 -cell INV1 -pg_pin_name VDD
```

SEE ALSO

calculate_rail_voltage(2)
update_currents(2)
set_supply_voltage(2)

set_voltage_regulator_lib_cell

Creates library cells for the voltage regulator cell masters.

SYNTAX

```
status set_voltage_regulator_lib_cell
   cell_master_name_list
   -input_power_pin pin_name voltage
   -output_power_pin pin_name voltage
   -ground_pin pin_name voltage
   -reference_pin pin_name voltage
   [-other_io_pin pin_name voltage]
   [-dropout_voltage dropout_voltage_value]
   [-output_voltage_variation_curve list_of_current_voltage_values]
```

Data Types

```
list cell_master_name_list
list pin_name voltage
float dropout_voltage_value
list list_of_current_voltage_values
```

ARGUMENTS

cell_master_name_list

Specifies the names of the cell masters for which the command creates the library cells.

-input_power_pin *pin_name voltage*

Specifies the names and voltage values for the input power pins.

-output_power_pin *pin_name voltage*

Specifies the names and voltage values for the output power pins.

-ground_pin *pin_name voltage*

Specifies the names and voltage values for the ground pins.

-reference_pin *pin_name voltage*

Specifies the names and voltage values for the reference pins.

-other_io_pin *pin_name voltage*

Specifies the names and voltage values for other input or output pins.

-dropout_voltage *dropout_voltage value*

Specifies the dropout voltage value for the voltage regulator cell master.

-output_voltage_variation_curve *list_of_current_voltage_values*

Specifies the output voltage variation curve for the voltage regulator cell master.

DESCRIPTION

This command creates an in-memory library cell for a voltage regulator cell master which does not exist in the library. Execute the command before the **open_mw_cel** command, or the command run will fail. When the creation process is complete, the command saves the created library cell in the rail_sidefile_lib_0 library.

To examine the library information for the newly created library cell, use the **report_voltage_regulator_lib_cell** command.

EXAMPLES

The following example creates a voltage regulator cell for the cell master VR1, and then reports the content of the newly created library cell VR1 with the **report_voltage_regulator_lib_cell** command.

```
pr_shell> set_voltage_regulator_lib_cell VR1 \
            -input_power_pin {VCC 3.0} -output_power_pin {VDD 1.3} \
            -reference_pin {REF 1.3} -ground_pin {VSS 0.0} \
            -other_io_pin {B 3.0}
1
pr_shell> report_voltage_regulator_lib_cell

****************************************
Report : voltage regulator library cells
Design :
****************************************

  Lib cell       Input power    Output power   Ground         Reference      Other io
                 pin            pin            pin            pin            pin
-------------------------------------------------------------------------------
  VR1            VCC (3V)       VDD (1.3V)     VSS (0V)       REF (1.3V)     B (3.0V)
1
```

SEE ALSO

report_voltage_regulator_lib_cell(2)

set_well_capacitance

Assigns well capacitances to the library cells.

SYNTAX

```
status set_well_capacitance
   [-unit_length unit_length_cap]
   [-unit_area unit_area_cap]
   object_list
```

Data Types

```
float unit_length_cap
float unit_area_cap
list object_list
```

ARGUMENTS

-unit_length *unit_length_cap*

Specifies the unit length well capacitance value to be assigned, in the library units. A cell's well capacitance is calculated as unit length capacitance multiplied by the cell's perimeter of the bbox. When a cell has multiple power and ground pins (including bias pins), this cell well capacitance is then evenly distributed to all power and ground pins.

-unit_area *unit_area_cap*

Specifies the unit area well capacitance value to be assigned, in the library units. Well capacitance is calculated as unit area capacitance multipled by the cell's area of the bbox. When a cell has multiple power and ground pins (including bias pins), this cell well capacitance is then evenly distributed to all power and ground pins.

object_list

Specifies a list of library cells on which to set the well capacitance value.

DESCRIPTION

The **set_well_capacitance** command assigns well capacitances to the library cells. All cell instances of the *lib_cell* have the specified well capacitance applied. The command returns 1 if successful, and 0 otherwise.

The -unit_length and -unit_area options are mutually exclusive. By default, the given well capacitance is evenly distributed to all the power and ground pins of the cell instances.

Run the **set_well_capacitance** command to set well capacitances before executing the **update_currents** command. During the run of the **update_currents** command, the tool first calculates intrinsic parasitics based on the information in the Synopsys libraries, and then combines the results with the user-assigned well capacitance value to generate the final intrinsic RC value for rail analysis. Therefore, during rail analysis, the combined capacitance will be used.

If user has issued **set_intrinsic_rc** command to explicitly assign intrinsic RC for a library cell or a cell instance, PG pin based well capacitance will not be added to the user assigned intrinsic RC. As a result, during rail analysis, only used assigned intrinsic RC will be used.

The PG pin's *intrinsic_rc* or *well_capacitance* attributes can be used to query intrinsic RC or well capacitance values respectively.

To overwrite the final intrinsic RC value, use the **set_intrinsic_rc** command.

EXAMPLES

The following example assigns the unit length well capacitances to a library cell.

```
pr_shell> set_well_capacitance -unit_length 1.0 [get_lib_cell lib1/INV]
pr_shell> update_currents -dynamic
```

The following example assigns the unit area well capacitances to a library cell, then query well capacitance from one instance:

```
pr_shell> set_well_capacitance -unit_area 2.0 lib2/XOR
pr_shell> get_attribute [get_cell xor_ins1] well_capacitance
```

SEE ALSO

update_currents(2)
set_intrinsic_rc(2)
get_attribute(2)

set_wire_load_min_block_size

Sets the minimum block area for automatic wire load selection. Any blocks with an area below the minimum are promoted to the minimum.

SYNTAX

```
int set_wire_load_min_block_size
   block_size
```

Data Types

block_size	float

ARGUMENTS

block_size

Minimum block size for auto wire load selection. This float value must be greater than or equal to 0.0.

DESCRIPTION

Specifies the minimum block area for automatic wire load selection in the current design. When automatic wire load selection is enabled, hierarchical cells are assigned wire load models based on their area. Any cells with an area below the minimum block size are assigned the minimum *block_size* value before a wire load model is selected. If the minimum block size is not explicitly set, the default value of 0.0 used.

EXAMPLES

This example specifies a minimum block size of 1000 area units.

```
pt_shell> set_wire_load_min_block_size 1000
```

SEE ALSO

report_wire_load(2)

```
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
```

set_wire_load_mode

Sets wire load mode for the current design.

SYNTAX

```
string set_wire_load_mode
   mode_name
```

Data Types

mode_name	string

ARGUMENTS

mode_name

Name of mode: *top*, *enclosed*, or *segmented*.

DESCRIPTION

Specifies a wire load mode with the **set_wire_load_mode** command. If the mode for the top level design is *top*, the wire load model on hierarchical cells has no effect, and the top-level wire load model is used to compute wire capacitance for all nets within the design.

If the mode for the top-level design is either *enclosed* or *segmented*, wire load models on hierarchical cells are used to calculate wire capacitance, resistance, and area in nets inside these blocks. If the *enclosed* mode is set, net values are determined using the wire load model of the hierarchical cell which fully encloses the net. If the *segmented* mode is set, segments of the net in each level of hierarchy are calculated separately, and the net value is the total of all segments. Hierarchical boundary pins are included in the pin count for the segment.

If no **wire_load_model_mode** is specified for a design, a default wire load mode is searched for in the first library in the link path. If the library checked does not have a default set, *top* is assumed.

EXAMPLES

This example sets the wire load mode to *enclosed* on the current design.

```
pt_shell> set_wire_load_mode enclosed
```

SEE ALSO

```
report_wire_load(2)
set_wire_load_model(2)
```

set_wire_load_model

Sets wire load model on designs, ports, or hierarchical cells.

SYNTAX

```
int set_wire_load_model
    -name model_name
   [-library lib_spec]
   [-min]
   [-max]
   [object_list]
```

Data Types

```
model_name       string
lib_spec         string
object_list      list
```

ARGUMENTS

-name *model_name*

Specifies the name of the wire load model; must be a valid wire load model in a library specified in the **link_path** command.

-library *lib_spec*

Specifies the library in which to search for the *model_name* variable. If not specified, libraries in the **link_path** command are searched.

-min

Indicates that the model is for minimum conditions.

-max

Indicates that the model is for maximum conditions.

object_list

Specifies a list of designs, ports, or hierarchical cells. The default is to set the wire model on the current instance, if set; or on the current design otherwise.

DESCRIPTION

Sets the wire load model on designs, ports, or hierarchical cells. The wire load model calculates net capacitance, resistance, and area for designs that have not been placed and routed.

Use automatic wire load selection for the design as another way to specify wire load models. If this is enabled, hierarchical blocks have wire load models assigned automatically based on their cell area.

To remove the wire load model setting, use the **remove_wire_load_model** command.

EXAMPLES

The following example specifies a wire load model of "BIG" on the top level design, a model of "MEDIUM" on hierarchical cell "u1", and a model of "SMALL" on hierarchical cell u1/u2.

```
pt_shell> current_design TOP
{"TOP"}
pt_shell> set_wire_load_model -name BIG
1
pt_shell> current_instance u1
u1
pt_shell> set_wire_load_model -name MEDIUM
1
pt_shell> current_instance u2
u1/u2
pt_shell> set_wire_load_model -name SMALL
1
pt_shell> current_instance
Current instance is the top-level of design 'TOP'.
pt_shell>
```

The following example specifies a wire load model of "BIG" on the top-level design, and a model of "SMALL" on all instances of design "adder."

```
pt_shell> current_design TOP
{"TOP"}
pt_shell> set_wire_load_model -name BIG
1
pt_shell> set_wire_load_model -name SMALL [all_instances -hierarchy adder]
1
```

The following example sets the external wire load model for ports OUT1* to 70x70, and specifies an external fanout number of 2.

```
pt_shell> set_wire_load_model 70x70 [get_ports OUT1*]
pt_shell> set_port_fanout_number 2 [get_ports OUT1*]
```

SEE ALSO

remove_wire_load_model(2)

```
report_lib(2)
report_wire_load(2)
set_port_fanout_number(2)
set_wire_load_mode(2)
set_wire_load_selection_group(2)
auto_wire_load_selection(3)
link_path(3)
```

set_wire_load_selection_group

Sets the wire load selection group for current design.

SYNTAX

```
int set_wire_load_selection_group
   [-min]
   [-max]
   [-library lib_spec]
   selection_group_name
   [object_list]
```

Data Types

```
lib_spec                  string
selection_group_name      string
object_list               list
```

ARGUMENTS

-min

Specifies the selection group for minimum conditions.

-max

Specifies the selection group for maximum conditions.

-library *lib_spec*

Specifies the library in which to search for the *selection_group_name* variable. If it is not specified, libraries in the **link_path** command are searched.

selection_group_name

Name of the selection group. This must be a valid selection group in a library specified in the **link_path** command.

object_list

Provides a list of hierarchical cells or designs. If you do not specify the *object_list* variable, the wire load selection group is set on the current instance, or on the current design if current instance is not set.

DESCRIPTION

Specifies the name of the selection group used when determining the wire load model based on cell area of blocks, when the **auto_wire_load_selection** command is true. This option is required only when the main library has more than one selection group defined, and the default group defined in the library is not the desired group. Automatic wire load selection is supported only for the *enclosed* wire load mode, specified using the **set_wire_load_mode** command.

To remove the wire load selection group setting, use the **remove_wire_load_selection_group** command.

EXAMPLES

This example specifies that the selection group named "selgrp1" from library "tech_lib" be applied to the current design.

```
pt_shell> set_wire_load_selection_group selgrp1 -library tech_lib
```

SEE ALSO

```
remove_wire_load_selection_group(2)
report_wire_load(2)
report_lib(2)
set_wire_load_min_block_size(2)
set_wire_load_mode(2)
set_wire_load_model(2)
auto_wire_load_selection(3)
link_path(3)
```

setenv

Sets the value of a system environment variable.

SYNTAX

```
string setenv variable_name new_value
```

```
string variable_name
```

```
string new_value
```

ARGUMENTS

variable_name

Names of the system environment variable to set.

new_value

Specifies the new value for the system environment variable.

DESCRIPTION

The **setenv** command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The **setenv** command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use **getenv** and **setenv**.

Environment variables are stored in the Tcl array variable **env**. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The **setenv** command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the **exec** command after a usage of **setenv** inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

EXAMPLES

The following example changes the default printer.

```
shell> getenv PRINTER
laser1
shell> setenv PRINTER "laser3"
laser3
shell> getenv PRINTER
laser3
```

SEE ALSO

exec(2)
getenv(2)
unsettenv(2)
printenv(2)
printvar(2)
set(2)
sh(2)
unset(2)

sh

Executes a command in a child process.

SYNTAX

string **sh** [*args*]

string *args*

ARGUMENTS

args

Command and arguments that you want to execute in the child process.

DESCRIPTION

This is very similar to the **exec** command. However, file name expansion is performed on the arguments. Remember that quoting and grouping is in terms of Tcl. Arguments which contain spaces will need to be grouped with double quotes or curly braces. Tcl special characters which are being passed to system commands will need to be quoted and/or escaped for Tcl. See the examples below.

EXAMPLES

This example shows how you can remove files with a wildcard.

```
prompt> ls aaa*
aaa1      aaa2      aaa3
prompt> sh rm aaa*
prompt> ls aaa*
Error: aaa*: No such file or directory
       Use error_info for more info. (CMD-013)
```

This example shows how to grep some files for a regular expression which contains spaces and Tcl special characters:

```
prompt> exec cat test3.out
blah blah blah
blah blah blah c blah
```

```
input [1:0] A;
output [2:0] B;
prompt>
prompt> sh egrep -v {[ ]+c[ ]+} test3.out
blah blah blah
prompt>
prompt> sh egrep {t[ ]+\[} test3.out
input [1:0] A;
output [2:0] B;
```

SEE ALSO

exec(2)

size_cell

Relinks leaf cells to a new library cells that have the required drive strength or other properties.

SYNTAX

```
int size_cell
   [-current_library]
   [-libraries lib_spec]
   cell_list
   lib_cell
```

Data Types

lib_spec	list
cell_list	list
lib_cell	string

ARGUMENTS

-current_library

If this option is specified, PrimeTime searches for library cells only in the cells' current libraries. You cannot specify this option with the **-libraries** option. You cannot specify this option if a full library cell name has been specified.

-libraries *lib_spec*

If this option is specified, PrimeTime resolves library cells from the libraries contained only in the *lib_spec*. Libraries are searched in the order in which they appear in *lib_spec*. The *lib_spec* value can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the **get_libs** command. You cannot specify this option with the **-current_library** option. You cannot specify this option if a full library cell name has been specified.

cell_list

Specifies a list of leaf cells to be relinked. Each cell must be in scope at or below the current instance.

lib_cell

Specifies the name of the library cell to which the specified cells are to be linked. The *lib_cell* option can be a library cell object, or the name of a library cell. The former can be obtained using the **get_lib_cells** command. The latter can either be the full library cell name, for example *'lib_name/lcell_name'*, or the just the base name of the library cell, for example *'lcell_name'*. You cannot specify either the **-current_library** or **-libraries** option and explicitly specify the full library cell name. If you invoke PrimeTime with the **-multi_scenario** option, you must use only the library cell base name. For more information, see the section entitled "RESOLVING LIBRARY CELLS".

DESCRIPTION

This command changes the drive strength or other properties of a leaf cell by relinking it to a new library cell that has the required properties. Like all other netlist editing commands, for the **size_cell** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **size_cell** command returns a 1 if successful and a 0 if unsuccessful. Each cell in the *cell_list* must be in scope; that is, at or below the current instance.

You can get a list of library cells that are compatible with a given cell using the **get_alternative_lib_cells** command, or report on the effects of compatible library cells using the **report_alternative_lib_cells** command.

The **size_cell** command is only for leaf cells. To swap a hierarchical block for a different design, you must use the **swap_cell** command.

Executing **size_cell** typically triggers an incremental timing update. However, a full timing update is required if the resized cell is in the clock tree or connected to the clock tree.

EQUIVALENT LIBRARY CELLS

Two library cells are compatible for sizing if all of the following restrictions are satisfied. The exact same set of restrictions applies to the **size_cell**, **get_alternative_lib_cells** and **report_alternative_lib_cells** commands.

- The library cells must be functionally equivalent. At least one of the following must be true, and none can be false:
  - If defined, the **function_id** attribute must match. This attribute is created by Library Compiler and is available on **lib_cell** objects.
  - If defined, the **mog_func_id** attribute must match. This attribute is created by Library Compiler for multiple output gate (MOG) cells and is available on **lib_cell** objects.
  - If defined, the **user_function_class** attribute must match. This is a user-defined functional equivalency attribute for cells that are too complex for Library Compiler to compute functional information for. This attribute is available on **lib_cell** objects.
- The number of pins, ordering of pins, pin directions, and internal pin names must match.
- The number of enabled library cell arcs, arc types and sense must match, including any conditional or "when" expressions.

RESOLVING LIBRARY CELLS

If the *lib_cell* option has been specified in base name only format, that is, without a library from which to resolve it from, PrimeTime resolves the library cell according to the following methodology.

If the **-current_library** option is specified, PrimeTime searches for library cells only in the cells' current libraries. If the **-libraries** option is specified, PrimeTime searches for library cells only in the libraries contained within the *lib_spec*.

Alternatively, if neither of the above options are specified, PrimeTime searches for the library cell in the following sources in this order:

1. The cell's current library.

2. The **link_path_per_instance** variable.
3. The **link_path** variable, if no **link_path_per_instance** specification pertains to the cell.

The first library cell that is found is used.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the **write_changes** man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

In the following example, an attempt to size a cell fails because the target is not functionally equivalent. After finding a compatible library cell, the second attempt succeeds.

```
pt_shell> size_cell o_reg1 class/NR4P
Error: Could not size 'o_reg1' ('class/FD2') with 'class/NR4P':
```

```
       Cells not functionally equivalent. (NED-005)
Error: No changes made. (NED-040)
0

pt_shell> get_alternative_lib_cells o_reg1
{"class/FD2P"}

pt_shell> size_cell o_reg1 class/FD2P
Information: Sized 'o_reg1' with 'class/FD2P'. (NED-045)
1
```

In the following example, cells are sized to alternative library cells using only the cells' current libraries and the library cell base name.

```
pt_shell> get_lib_cells -of_objects n1
{"class/ND2"}

pt_shell> get_alternative_lib_cells -current_library  n1
{"class/ND2P"}

pt_shell> get_lib_cells -of_objects n2
{"libA/ND2"}

pt_shell> get_alternative_lib_cells -current_library  n2
{"libA/ND2P"}

pt_shell> size_cell -current_library {n1 n2} ND2P
Information: Sized 'n1' with 'class/ND2P'. (NED-045)
Information: Sized 'n2' with 'libA/ND2P'. (NED-045)
1
```

In the following example, cells are sized to alternative library cells using only user-specified libraries and the library cell base name. Assume that there are no 'ND2P' library cells in 'lib1'. The 'lib1' library is searched first as it appears in the *lib_spec* list first and then 'lib2' is searched.

```
pt_shell> size_cell -libraries {lib1 lib2} {n1 n2} ND2P
Information: Sized 'n1' with 'lib2/ND2P'. (NED-045)
Information: Sized 'n2' with 'lib2/ND2P'. (NED-045)
1
```

SEE ALSO

```
get_alternative_lib_cells(2)
report_alternative_lib_cells(2)
swap_cell(2)
write_changes(2)
link_path_per_instance(3)
link_path(3)
```

sizeof_collection

Returns the number of objects in a collection.

SYNTAX

```
int sizeof_collection
   collection1
```

Data Types

collection1	collection

ARGUMENTS

collection1

Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

DESCRIPTION

The **sizeof_collection** command is an efficient mechanism for determining the number of objects in a collection.

EXAMPLES

The following example from PrimeTime shows a simple way to find out how many objects matched a particular pattern and filter in the **get_cells** command.

```
pt_shell> echo "Number of hierarchical cells: \
?                 [sizeof_collection \
?                   [filter_collection [get_cells * -hier \
?                     "is_hierarchical == true"]]]"
Number of hierarchical cells is: 10
```

The following example from PrimeTime shows what happens when the argument for the **sizeof_collection** command results in an empty collection.

```
pt_shell> set s1 [get_cells *]
{"u1", "u2", "u3"}
```

```
pt_shell> set ssize [filter_collection $s1 "area < 0"]
pt_shell> echo "Cells with area < 0: [sizeof_collection $ssize]"
Cells with area < 0: 0
pt_shell> unset s1
```

SEE ALSO

collections(2)
filter_collection(2)

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

SYNTAX

```
collection sort_collection
   [-descending]
   [-dictionary]
   collection
   criteria
```

Data Types

collection	collection
criteria	list

ARGUMENTS

-descending

Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

-dictionary

Sort strings dictionary order. For example "a30" would come after "a4".

collection

Specifies the collection to be sorted.

criteria

Specifies a list of one or more application or user-defined attributes to use as sort keys.

DESCRIPTION

You can use the **sort_collection** command to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the *is_hierarchical* and *full_name* attributes as *criteria*.

The *criteria* can specify an attribute on another object. The other object must be available as an attribute on this object. For example, if you had a collection of net shapes, you can sort the objects by net using *owner_net.name*

for instance.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. The *-descending* option reverses the order of the objects.

EXAMPLES

The following example from PrimeTime sorts a collection of cells based on hierarchy, and adds a second key to list them alphabetically. In this example, cells i1, i2 and i10 are hierarchical, and o1 and o2 are leaf cells. Because the *is_hierarchical* attribute is a Boolean attribute, those objects with the attribute set to *false* are listed first in the sorted collection.

```
pt_shell> set zc [get_cells {o2 i2 o1 i1 i10}]
{"o1", "i2", "o1", "i1", "i10"}
pt_shell> set zsort [sort_collection $zc {is_hierarchical full_name}]
{"o1", "o2", "i1", "i10", "i2"}
pt_shell> set zsort [sort_collection -dictionary $zc {is_hierarchical full_name}]
{"o1", "o2", "i1", "i2", "i10"}
```

SEE ALSO

collections(2)

source

Read a file and evaluate it as a Tcl script.

SYNTAX

`string` **`source`** `[-echo] [-verbose] [-continue_on_error]` *`file`*

`string` *`file`*

ARGUMENTS

-echo

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

-continue_on_error

Don't stop script on errors. Similar to setting the shell variable sh_continue_on_error to true, but only applies to this particular script.

file

Script file to read.

DESCRIPTION

The **source** command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the **source** command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the **-echo** and **-verbose** options. The **-echo** option echoes each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

NOTE: To emulate the behavior of the dc_shell **include** command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for based only on what you typed. However, if the system variable sh_source_uses_search_path is set to "true", the file is searched for based on the path established with the search_path variable.

The **source** command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. The **-echo** and **-verbose** options are ignored for gzip formatted files.

EXAMPLES

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

SEE ALSO

search_path(3)
sh_source_uses_search_path(3)
sh_continue_on_error(3)

start_hosts

Starts the hosts specified by the **set_host_options** command.

SYNTAX

```
status start_hosts
   [-timeout   seconds]
   [-min_hosts num_hosts]
```

Data Types

```
seconds       integer
num_hosts     integer
```

ARGUMENTS

-timeout *seconds*

Specifies, in seconds, how long the **start_hosts** command waits for the remote hosts to come online. While the command is waiting, no other commands can be executed at the master. If all the remote hosts come online before the timeout has expired, the command returns immediately on connection of the final host. If all the remote hosts do not come online before the timeout has expired, the command returns and accept the remainder of the hosts online in the background. The value specified must be greater than or equal to zero. If the option is not specified, the default is 21600s (6 hours).

-min_hosts *num_hosts*

Specifies the minimum number of hosts that the **start_hosts** command waits to come online. While the command is waiting, no other commands can be executed at the master. As soon as the minimum number of hosts have come online, the command returns immediately and allow commands to execute at the master. The **-timeout** option to the command overrides the **-min_hosts** option. Therefore, if the minimum number of hosts does not come online before the timeout has expired, the command is governed by the behavior of the **-timeout** option as described previously. The value specified must be greater than or equal to zero. If the option is not specified, by default the command waits for all hosts launched to come online.

DESCRIPTION

This command explicitly starts the hosts to be used as compute resources. Host options must first be setup using the **set_host_options** command. If the hosts are not explicitly started using this command, they are started during execution of the **update_timing** command.

When hosts are started they end up in one of the following states

```
LAUNCHED      : The process is launched but not ready for use yet.
ONLINE        : The process is online and ready for use.
SHUTDOWN      : The process was shutdown.
FATAL         : The process abnormally terminated.
MISALIGNED    : The master and slave processes are different versions
                of PrimeTime and the slave process is not usable.
SHUTTING_DOWN : The process is shutting down.
```

Remote Connection Considerations

1. For the master process to interact with remote processes, there is a connection between the master and the remote host. Each connection consumes two file descriptors from the operating system. Generally, there is a limit on the number of file descriptors that a user-specified process can open simultaneously. Typically, a user-specified process is allowed to open 1024 simultaneous file descriptors. Because the master PrimeTime process uses several descriptors for normal operations, a maximum of 500 remote processes can be launched. The **start_hosts** command rejects any hosts beyond 500. To see the system user limit on the number of file descriptors allowed, run the **limit descriptors** command in the UNIX terminal.

2. When using the **rsh** command to launch remote processes, depending on the host configuration, there can be imposted limits in relation to the number of allowed connections.

   - The host that you want to connect to can define a limit on the number of simultaneous connections that a process can make to the host in the /etc/xinetd.conf file. The **instances** keyword defines the maximum number of connections allowed. After the number of connections exceeds the limit, the circuit will fail to setup, and a protocol failure will be returned as an error. If this occurs, the master log file shows the following message:
     poll: protocol failure in circuit setup

     Action: The system limit on the number of incoming connections can be changed only by a superuser. However, you can specify fewer remote processes to be launched on the specific remote host than the instance limit specified by that remote host.

   - The host launching the remote processes can define a limit on the range of privileged ports accessible to you. The range is typically 512-1024. After all privileged ports are exhausted, subsequent **rsh** connections will fail to set up, and an out of sockets error will be returned as an error. If this occurs, the master log file shows the following message:
     rcmd: socket: All ports in use

     Action: The system limit on the number of privileged ports available can be changed only by a superuser. However, you can use a different protocol to launch the subsequent processes beyond the number of privileged ports. For example, **ssh** does not use privileged ports, or you can submit the remote process to a managed resource such as LSF or Grid.

EXAMPLES

To start and verify the hosts, use the following flow:

```
# Specify the options of each host
set_host_options -name options1 ...
set_host_options -name options2 ...
...

# Start the hosts
start_hosts ...

# Verify that the hosts started correctly and report the usage
```

```
report_host_usage -verbose
```

SEE ALSO

```
remove_host_options(2)
report_host_usage(2)
set_host_options(2)
stop_hosts(2)
```

start_icc2_gui

Starts the ICC2 GUI.

SYNTAX

```
string start_icc2_gui
   [-icc2_shell_path icc2_shell_path]
```

Data Types

```
string icc2_shell_path
```

ARGUMENTS

-icc2_shell_path *icc2_shell_path*

Specifies the full path to where icc2_shell resides. Using this option overrides the $PATH environment variable and skips version check.

DESCRIPTION

This command starts the ICC2 GUI from the pr_shell prompt. The command is ignored if the ICC2 GUI has already been started and hasn't exited.

By default, **start_icc2_gui** searches the $PATH environment variable for icc2_shell and chooses the first one.

start_icc2_gui performs version check on the chosen icc2_shell.

If version check succeeds, **start_icc2_gui** uses the chosen icc2_shell to launch an ICC2 GUI with the current PrimeRail search_path, design, rail result, and taps, whichever is available.

If version check fails, the command aborts.

EXAMPLES

The following example starts the ICC2 GUI.

```
pr_shell> start_icc2_gui
```

SEE ALSO

`current_rail_result(2)`

start_profile

Starts profiling PrimeTime commands.

SYNTAX

```
status start_profile
  [-output directory_name]
  [-verbose]
  [-include report_list]
```

Data Types

```
directory_name     string
report_list        list
```

ARGUMENTS

-output *directory_name*

Specifies the name of a directory to save the profiling reports to. By default, the output directory name is *profile* in the current working directory.

-verbose

Specifies the tracking of all commands in profiling reports. Without this option, only commands which take more than 0.1 second to run or use more than 100 Kb of memory are reported.

-include *report_list*

Limits the profiling actions to selected items. Allowable list entries are: *stopwatch*, *tcl_profile* and *perf_cpu*.

DESCRIPTION

The **start_profile** command initiates command-level profiling in PrimeTime.

For every PrimeTime command or procedure invocation, the cpu time, elapsed time, delta memory, and number of calls are registered. Several commands or procedure invocations at the same scope level are collapsed into one and their CPU times, elapsed times, and delta memory are added; the number of calls is increased by one.

The PrimeTime profile handles the source command in a special manner. Since source can be used to arbitrarily call deep levels of scripts, source commands also show file names. Therefore, for source commands, even if two source commands are executed at the same scope level, they usually have two different entries, unless they source the exact same file.

In addition to the PrimeTime profile output, a stopwatch report is also created. For each PrimeTime command, a start and an end **events** are recorded. At each stopwatch event, metrics such as current CPU time, elapsed time, and peak memory usage are saved. For each metric, a difference between the current and the previous value is also reported.

PrimeTime profiling reports are written to an output directory at the exit of a PrimeTime session or when the command **stop_profile** is executed. Stopping of profiling also creates a summary HTML page with links to profiling reports or data files with short descriptions.

Use **-include** options to select only the report you want to see. For example, if you require only the code-level performance profiling data, use "**-include {perf_cpu}**".

The **stop_profile** command is used to stop the command level profiling in PrimeTime that was previously started by the **start_profile** command.

In the distributed multi-scenario flow, use **remote_execute** command to activate profiling of remote processes. Profiling is performed on a PrimeTime process, not on a scenario. So if there are fewer hosts than scenarios, the profiling output appears only in scenario output directories for scenarios that were first to execute on worker hosts.

EXAMPLES

In the following script example of profiling, the reports are written to the reports/run_profile directory .

```
start_profile -output reports/run_profile
 ...   # Usual PT script
stop_profile
```

The following script example shows how to enable Tcl profiling in distributed multi- scenario analysis (DMSA). The profiling is first enabled on the DMSA master and then on DMSA workers.

```
set multi_scenario_working_directory ./ms_data
set master_profile_dir $multi_scenario_working_directory/master_profile
start_profile -output $master_profile_dir
 ...    # Setting up DMSA

current_session -all
remote_execute {
  set outdir $multi_scenario_working_directory/[current_scenario]/worker_profile
  start_profile -output $outdir
}
```

SEE ALSO

```
current_scenario(2)
remote_execute(2)
stop_profile(2)
```

stop_hosts

Stops all hosts that have been started.

SYNTAX

```
int stop_hosts
```

ARGUMENTS

None.

DESCRIPTION

The **stop_hosts** command shuts down all hosts that have been started. The hosts could have been explicitly launched using the **start_hosts** command or implicitly launched during the execution of the **update_timing** command.

EXAMPLES

In the following example, the master process is running on the host named ptnlm15 using a 64-bit binary and specifies several different sets of host options.

The first host options, named "my_opts1", specifies

- to use one process on the same host as the master
- no upper limit on the number of cores to use so the default applies.

The second host options, named "my_opts2", specifies

- to use one process on the same host as the master, explicitly mentioning the master host by name
- to use a maximum of two cores.

The third host options, named "my_opts3", specifies

- to use one process on the host named ptnlm11, since no submit command is specified, "rsh" will be used to connect to ptnlm11
- to use a maximum of three cores.

The fourth host options named "my_opts4", specifies

- to use one process on the host named ptnlm12 connecting to ptnlm12 using the submit command "/usr/bin/rsh"
- no upper limit on the numebr of cores to use so the default applies.

The fifth host options named "my_opts5", specifies

- to use one process on an LSF farm, using the "bsub" command found from the environment path to submit the job, requiring 1GB of memory and two slots per compute server. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use with the **-n 2 -R span\[ptile=2\]** options.
- to use the terminate command "bkill" found from the environment path for the termination of jobs that do not come online.
- to use a maximum of two cores.

The sixth host options named "my_opts6", specifies

- to use one process on a Grid farm, using the "qsub" command found from the environment path to submit the job, using the project named "bnormal", requiring 1GB of memory and four slots per compute server. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use by targeting the parallel environment "mt" requesting 4 slots with the "-pe mt 4" options.
- to use the terminate command "qdel" found from the environment path for the termination of jobs that do not come online.
- to use a maximum of four cores.

Note: Before calling the **stop_hosts** command, all the processes report as ONLINE and after calling the **stop_hosts** command all the processes report as SHUTDOWN. The order in which the hosts shutdown is random.

Call the **set_host_options** command to specify the host options.

```
pt_shell> set_host_options -name my_opts1 -num_processes 1
1
pt_shell> set_host_options -name my_opts2 -num_processes 1 \
          ptnlm15 -max_cores 2
1
pt_shell> set_host_options -name my_opts3 -num_processes 1 \
          ptnlm11 -max_cores 3
Warning: no submit command specified to access remote host
 'ptnlm10'. The 'rsh' command will be used. (CMCR-004)
1
pt_shell> set_host_options -name my_opts4 -num_processes 1 \
          -submit_command "/usr/bin/rsh" ptnlm12
1
pt_shell> set_host_options -name my_opts5 -num_processes 1 \
          -submit_command "[sh which bsub] -n 2 -R rusage\[mem=1000\] \
          -R span\[ptile=2\]" -terminate_command "[sh which bkill]" \
          -max_cores 2
1
pt_shell> set_host_options -name my_opts6 -num_processes 1 \
          -submit_command "[sh which qsub] -P bnormal -pe mt 4 \
          -l mem_free=1G" -terminate_command "[sh which qdel]" \
          -max_cores 4
1
```

Call the **start_hosts** command to start the slave processes.

```
pt_shell> start_hosts
Launching 6 Distributed Workers

   1] Launched : /J-2014.12/bin/pt_shell \
```

```
                 -slv_type dmsa_slv -max_cores 4

  2] Launched : ptnlm15 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 2

  3] Launched : rsh ptnlm11 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 3

  4] Launched : /usr/bin/rsh ptnlm12 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 4

  5] Launched : /lsf/bin/bsub -n 2 -R rusage[mem=1000] -R span[ptile=2] \
                 /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 2

  6] Launched : /remote/sge2/default/bin/lx-amd64/qsub -P bnormal \
                 -pe mt 4 -l mem_free=1G /J-2014.12/bin/pt_shell \
                 -slv_type dmsa_slv -max_cores 4

 ---------------------------------------------------------------------------
 Distributed farm creation timeout : 21600 seconds
 Waiting for   6 (of   6) distributed hosts (Tue Nov 25 04:39:34 2014)
 Waiting for   2 (of   6) distributed hosts (Tue Nov 25 04:39:49 2014)
 Waiting for   0 (of   6) distributed hosts (Tue Nov 25 04:39:59 2014)
 ---------------------------------------------------------------------------

1
```

Call the **report_host_usage** command after starting the hosts to report the host options specified by the **set_host_options** command, as well as the real time data associated with the processes.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 04:59:46 2014
****************************************

  Options Name          Host Name              Num Processes
  ---------------------------------------------------------------------
  my_opts1              **localhost**          1
  my_opts2              **ptnlm15**            1
  my_opts3              ptnlm11                1
  my_opts4              ptnlm12                1
  my_opts5              >>farm<<               1
  my_opts6              >>farm<<               1

  Options Name     #    Host Name    Job ID    Process ID    Status
  ---------------------------------------------------------------------
  my_opts1         1                 27337     27337         ONLINE
  my_opts2         2    ptnlm15      27378     27378         ONLINE
  my_opts3         3    ptnlm11      23716     23716         ONLINE
  my_opts4         4    ptnlm12      13059     13059         ONLINE
  my_opts5         5    maddog136    312487    21901         ONLINE
  my_opts6         6    peemt32      2165476   5293          ONLINE

  Usage limits (cores)

  Options Name     #                                         Effective
  ---------------------------------------------------------------------
  (local process)  -                                         4
  my_opts1         1                                         -
  my_opts2         2                                         -
  my_opts3         3                                         -
```

```
  my_opts4         4                                        -
  my_opts5         5                                        -
  my_opts6         6                                        -
  ---------------------------------------------------------------------
  Total                                                     4

  Memory usage

  Options Name     #                           Memory (MB)
  ---------------------------------------------------------------------
  (local process)  -                           445.43
  my_opts1         1                           0.00
  my_opts2         2                           0.00
  my_opts3         3                           0.00
  my_opts4         4                           0.00
  my_opts5         5                           0.00
  my_opts6         6                           0.00

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ---------------------------------------------------------------------
  (local process)  -    2                      1023
  my_opts1         1    0                      0
  my_opts2         2    0                      0
  my_opts3         3    0                      0
  my_opts4         4    0                      0
  my_opts5         5    0                      0
  my_opts6         6    0                      0

1
```

Call the **stop_hosts** command so shutdown the processes.

```
pt_shell> stop_hosts
Shutting down all slaves processes ...
 Shutdown Process 1
 Shutdown Process 2
 Shutdown Process 3
 Shutdown Process 4
 Shutdown Process 5
 Shutdown Process 6
1
```

Call the **report_host_usage** command after stopping the hosts to report the host options created by the **set_host_options** command.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 05:00:00 2014
****************************************

  Options Name          Host Name              Num Processes
  ---------------------------------------------------------------------
  my_opts1              **localhost**          1
  my_opts2              **ptnlm15**            1
  my_opts3              ptnlm11                1
  my_opts4              ptnlm12                1
  my_opts5              >>farm<<               1
  my_opts6              >>farm<<               1

  Usage limits (cores)
```

```
Options Name     #                                          Effective
-------------------------------------------------------------------
(local process)  -                                          4
my_opts1                                                    -
my_opts2                                                    -
my_opts3                                                    -
my_opts4                                                    -
my_opts5                                                    -
my_opts6                                                    -
-------------------------------------------------------------------
Total                                                       4

Memory usage

Options Name     #                           Memory (MB)
-------------------------------------------------------------------
(local process)  -                           445.43

Performance

Options Name     #    CPU Time (s)           Elapsed Time (s)
-------------------------------------------------------------------
(local process)  -    2                      1023

1
```

SEE ALSO

start_hosts(2)
set_host_options(2)
remove_host_options(2)
report_host_usage(2)

stop_profile

Stop profiling PrimeTime commands.

SYNTAX

status **stop_profile**

ARGUMENTS

None.

DESCRIPTION

The **stop_profile** command is used to stop the command level profiling in PrimeTime that you started previously using the **start_profile** command. You should use this command in conjunction with the **start_profile** command to write out profiling information for PrimeTime scripts.

Stopping a profile stops collecting information about commands for profiling and writes out profiling reports into the directory specified by **start_profile -output**. By default, the output directory name is *profile* in the current working directory.

This command succeeds all the time except when the profiling is not currently active.

EXAMPLES

The following gives an example of profiling.

```
pt_shell> start_profile
pt_shell> ....   # Usual PT script
pt_shell> stop_profile
```

SEE ALSO

```
start_profile(2)
```

suppress_message

Disables printing of one or more informational or warning messages.

SYNTAX

string **suppress_message** [*message_list*]

list *message_list*

ARGUMENTS

message_list

A list of messages to suppress.

DESCRIPTION

The **suppress_message** command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of **suppress_message** is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using **unsuppress_message**) as many times as it was suppressed in order for it to be enabled. The **print_suppressed_messages** command displays the currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
prompt> suppress_message CMD-029
prompt> unalias q*
prompt>
```

SEE ALSO

print_suppressed_messages(2)
unsuppress_message(2)
get_message_ids(2)
set_message_info(2)

swap_cell

Swaps one or more cells with a new design or library cell.

SYNTAX

```
integer swap_cell
   cell_list
   swap_in
   [-dont_preserve_constraints]
   [-file file_name]
   [-format db | verilog | vhdl]
```

Data Types

```
cell_list         list
swap_in           string
file_name         string
```

ARGUMENTS

cell_list

Specifies a list of cells to be swapped out.

swap_in

Specifies the name of the design or library cell to be swapped in.

-dont_preserve_constraints

Indicates that **swap_cell** is not to reapply the current design constraints after the swap.

-file *file_name*

Specifies the name of a file that contains a design that is to be swapped in.

-format db | verilog | vhdl

Specifies the format for *file_name*. Allowed values are **db** (the default), **verilog**, and **vhdl**.

DESCRIPTION

The **swap_cell** command replaces the cells in the *cell_list* option with *swap_in* (a new design or library cell), and incrementally relinks the design. The command checks to ensure that the pinouts of the swapped-in and swapped-

out cells are the same. For example, every boundary pin of the cell being swapped out must have a counterpart with the same name in the cell being swapped in, and vice versa.

The **swap_cell** command is considered a netlist editing command. Like all other netlist editing commands, for the **swap_cell** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **swap_cell** command returns a 1 if successful and a 0 if unsuccessful.

If you want to swap a library cell for another library cell that is just a different size, use the **size_cell** command instead of the **swap_cell** command. The **size_cell** command is more efficient than the **swap_cell** command, and it is optimized for incremental timing updates.

Library cells being swapped in must be in memory. Designs being swapped in can be in memory or can be retrieved from a file using the **-file** and **-format** options. For more information, see the "Interaction of Reading Netlists and swap_cell" section.

By default, design constraints are preserved by writing a script to the directory referenced by the **pt_tmp_dir** variable, then loading the script after the swap has completed. Use the **-dont_preserve_constraints** option only if you plan to perform several **swap_cell** commands; for example, in a loop. In this case, it is much more efficient to use the **write_script** command then the **swap_cell -dont_preserve_constraints** command, and then the **source** command in your script.

When a subdesign is swapped in or out using swap_cell, the annotated parasitics on nets connected to that cell are removed as they are no longer valid.

If a swapped cell is in the parent chain of any objects that are elements of a saved collection, these collections might be deleted based on the **collection_deletion_effort** variable. For more information, see the man pages for **collections** and associated variables.

The **swap_cell** command does not create black boxes. If the design that is swapped in contains unresolved references, the result of the swap is a partially linked design.

Interaction of Reading Netlists and swap_cell

The best way to swap in a design that is not in memory is to use the **-file** and **-format** options. When you load the new design this way, the **swap_cell** command maintains the current design, and the swap is a one-step operation. Explicitly reading a new design before swapping cells changes the current design, so you must remember to reset the current design back to the previous current design before using the **swap_cell** command. Thus, it is preferable to use the **-file** and **-format** options. Both alternatives are illustrated in the EXAMPLES section.

Interaction of Linking and swap_cell

Do not perform an explicit link after executing **swap_cell**. The **swap_cell** command implicitly relinks the part of the design you have changed; therefore, performing an explicit link after executing **swap_cell** can undo some or all of the work done by **swap_cell**.

For example, consider U2, an instance of design X, which contains instance U0, an instance of design Y. If you swap out U2/U0 for a different design, PrimeTime has modified the *instance* U2/U0; it has not modified the design Y. Therefore, an initial link causes U2/U0 to revert to the original Y (the version of Y inferred by the **link_path**).

PrimeTime does not save information about cells that have been swapped. That information persists only until the next link.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the **write_changes** command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the

write_changes man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with **swap_cell**. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by **link_design -remove_sub_designs**. Uniquified designs can be listed using **list_designs -all**. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the **size_cell** command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the **is_edited** Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the **is_edited** attribute on i1/low is **true** after the **size_cell** command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

EXAMPLES

The following example shows the optimal method for swapping in a design from a file. The design "A" is in "A.db". The **swap_cell** command replaces cells u1, u2, and u3 in TOP with the design A (in A.db).

```
pt_shell> current_design
{"TOP"}
pt_shell> swap_cell {u1 u2 u3} A -file A.db
Loading db file '/usr/designs/A.db'
 ...unlinking u1
 ...unlinking u2
 ...unlinking u3
1
pt_shell> current_design
{"TOP"}
```

The following example accomplishes the same task as the previous example, but shows the method where the design file is read explicitly.

```
pt_shell> current_design
{"TOP"}
pt_shell> read_db A.db
Loading db file '/usr/designs/A.db'
1
pt_shell> current_design
{"A"}
pt_shell> current_design TOP
{"TOP"}
pt_shell> swap_cell {u1 u2 u3} A.db:A
 ...unlinking u1
 ...unlinking u2
 ...unlinking u3
1
```

The following example replaces cells u1 and u2 in TOP with the lib_cell misc_cmos/LC3.

```
pt_shell> swap_cell {u1 u2} [get_lib_cells misc_cmos/LC3]
```

SEE ALSO

collections(2)
current_design(2)
link_design(2)
size_cell(2)
source(2)
write_changes(2)
write_script(2)
link_path(3)
pt_tmp_dir(3)

transform_exceptions

Performs transformation on timing exceptions.

SYNTAX

```
status transform_exceptions
   [-from from_list
      | -rise_from rise_from_list
      | -fall_from fall_from_list]
   [-through through_list
      | -rise_through rise_through_list
      | -fall_through fall_through_list]
   [-to to_list]
      | -rise_to rise_to_list
      | -fall_to fall_to_list]
   [-verbose]
   [-dry_run]
   [-remove_ignored reason_list]
   [-use_to_for_endpoints]
   [-flatten]
```

Data Types

```
from_list             list
rise_from_list        list
fall_from_list        list
through_list          list
rise_through_list     list
fall_through_list     list
to_list               list
rise_to_list          list
fall_to_list          list
reason_list           list
```

ARGUMENTS

-from *from_list*

Specifies a list of clocks, ports, cells, and pins in the current design. This list must specify valid path start points, such as primary inputs or register clock pins. The **-from**, **-rise_from**, and **-fall_from** options are mutually exclusive; you can use only one.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by the rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. The **-from**, **-rise_from**, and **-fall_from** options are mutually exclusive; you can use only one.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. The **-from**, **-rise_from**, and **-fall_from** options are mutually exclusive; you can use only one.

-through ***through_list***

Specifies a list of pins or ports. The **-through**, **-rise_through**, and **-fall_through** options are mutually exclusive; you can use only one.

-rise_through ***rise_through_list***

Specifies the same list as the **-through** option does, but applies only to paths that have a rising transition at the through points. The **-through**, **-rise_through**, and **-fall_through** options are mutually exclusive; you can use only one.

-fall_through ***fall_through_list***

Specifies the same list as the **-through** option does, but applies only to paths that have a falling transition at the through points. The **-through**, **-rise_through**, and **-fall_through** options are mutually exclusive; you can use only one.

-to ***to_list***

Specifies a list of clocks, ports, cells, and pins in the current design. This list must specify valid path endpoints, such as primary outputs or register D-pins. The **-to**, **-rise_to**, and **-fall_to** options are mutually exclusive; you can use only one.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. The **-to**, **-rise_to**, and **-fall_to** options are mutually exclusive; you can use only one.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. The **-to**, **-rise_to**, and **-fall_to** options are mutually exclusive; you can use only one.

-verbose

Provides a detailed output of each exception that exists on the set of paths specified by using the path selectors or, if none are specified, by using the entire design. Identifies whether the exception is completely valid, partially ignored, or not ignored in the context of the selected paths. Provides justification for ignoring exception paths from the following list:

- An exception path has an invalid start or endpoint.
- The multicycle path exception specifies single cycle values.
- There is a nonexistent path between exception specifiers.
- The exception is set on an unconstrained path.
- The exception specifies a path in the clock network.

- The exception is overridden by a higher precedence exception.
- The exception is overridden because of mode analysis.
- The exception is overridden due to exclusive clock groups.

-dry_run

Allows analysis and reporting, but precludes replacing the original exceptions with the transformed ones.

-remove_ignored ***reason_list***

Enables exception compaction to reduce ignored exceptions or trim exception specifiers to remove paths that ignore exception. Valid values for *reason_list* are **invalid_specifiers**, **no_path**, **unconstrained_path**, **clock_path**, and **overridden**.

-use_to_for_endpoints

Causes path endpoints specified using "-through*" to use "-to*" instead. This option fixes the transformation context of the command and therefore cannot be used in conjunction with the **-remove_ignored** or **-flatten** option.

-flatten

Converts user-compressed exceptions into simple exceptions such that each exception specifier **(-rise_from, -fall_from, -from, -rise_through, -fall_through, -through, -rise_to, -fall_to, -to)** matches a single object. This option fixes the transformation context of the command and therefore cannot be used in conjunction with the **-remove_ignored** or **-use_to_for_endpoints** option.

DESCRIPTION

Performs transformation on timing exceptions to determine the usage of each exception and interference between exceptions across the specified paths in the design. By default, the behavior of this command is to cleanup exceptions to reduce ignored exception specifiers within the context of specified design paths. The command outputs statistics about the processed set of exceptions. The **-verbose** option expands the output display usage and justification information about each exception processed.

The checks for invalid start or end point and the check for unconstrained path are performed only with respect to **(-rise_from, -fall_from, -from, -rise_to, -fall_to, -to)** exception specifiers.

The command can be run for reporting purposes only by using the **-dry_run** option. While all required analysis would still be performed as for the default case, the transformations are disregarded when the command completes.

To generate a file containing the new set of exceptions, first execute the command with the options you want, excluding the **-dry_run** option. Next, use the **write_script** or **write_sdc** commands to emit design constraints and context. Use the **-include** option with either **write_script** or **write_sdc** to emit exceptions only if you want them emitted.

EXAMPLES

The following example performs exceptions transformations on all timing exceptions set on the design.

```
pt_shell> transform_exceptions

****************************************
Report : transform_exceptions
Design : CORE
 ...
****************************************

Transformations Summary  false  multicycle  delay  Total
------------------------------------------------------------
  invalid specifiers     5      10          1      16
  non-existent paths     1      3           0      4
  clock network paths    0      1           0      1
  unconstrained paths    1      0           0      1
  overridden             0      2           1      3
------------------------------------------------------------
Total transformations                                25

Exceptions Summary       false  multicycle  delay  Total
------------------------------------------------------------
  Removed                0      2           1      3
  Modified               5      9           0      14
  Analyzed               7      16          1      24
```

The following example performs selective transformation on a specified set of paths without committing the final exceptions.

```
pt_shell> transform_exceptions -from {A B} -to D \
          -remove_ignored { overridden no_path } -dry_run -verbose

****************************************
Report : transform_exceptions
        -verbose
        -dry_run
 ...
****************************************

From          To          Setup              Hold
------------------------------------------------------------
{A B}         *           FALSE              FALSE

A             D           cycles=2           cycles=0
  OVERRIDDEN by false_path
        -from   A

B             D           cycles=3           cycles=0
  OVERRIDDEN by false_path
        -from   B


Transformations Summary  false  multicycle  delay  Total
------------------------------------------------------------
  overridden             0      2           0      2
------------------------------------------------------------
Total transformations                                2

Exceptions Summary       false  multicycle  delay  Total
------------------------------------------------------------
  Analyzed               1      2           0      3
```

SEE ALSO

```
report_exceptions(2)
set_clock_groups(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_mode(2)
set_multicycle_path(2)
write_script(2)
write_sdc(2)
```

unalias

Removes one or more aliases.

SYNTAX

```
string unalias
   patterns
```

ARGUMENTS

patterns

Specifies the patterns to be matched. This argument can contain more than one pattern. Each pattern can be the name of a specific alias to be removed or a pattern containing the wildcard characters ***** and **%**, which match one or more aliases to be removed.

DESCRIPTION

The **unalias** command removes aliases created by the **alias** command.

EXAMPLES

The following command removes all aliases.

```
prompt> unalias *
```

The following command removes all aliases beginning with f, and the alias rt100.

```
prompt> unalias f* rt100
```

SEE ALSO

alias(2)

unset_rtl_to_gate_name

Specifies the signals or objects that must never be mapped, or to undo the erroneously mapped signals or objects.

SYNTAX

```
status unset_rtl_to_gate_name
   name
```

Data Types

```
name      string
```

ARGUMENTS

name

Specifies the name of the RTL object that is not changed to the gate-level object by the name-mapping process.

DESCRIPTION

If you have done only the RTL simulation to generate the RTL backward SAIF or RTL VCD file, PrimeTime PX tries to do the name mapping to find the gate-level objects in the RTL SAIF or VCD file. To prevent the erroneous mapping of the RTL-to-gate-level objects, use the **unset_rtl_to_gate_name** command. This command unmaps the erroneous mapping applied to the specified RTL objects.

EXAMPLES

The following example unmaps the erroneously mapped RTL and gate-level objects. PrimeTime PX searches the **reg_i[1]** object in the gate-level design to see if it appears in the RTL VCD or SAIF file.

```
pt_shell> unset_rtl_to_gate_name reg_i[1]
```

SEE ALSO

```
set_rtl_to_gate_name(2)
```

unsetenv

Removes a system environment variable.

SYNTAX

```
string getenv
   variable_name
```

Data Types

variable_name	string

ARGUMENTS

variable_name

Specifies the name of the environment variable to be unset.

DESCRIPTION

The **unsetenv** command searches the system environment for the specified *variable_name* and removes variable from the environment. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **unsetenv**, commands is a convenience function to interact with this array. It is equivalent to 'unset ::env(*variable_name*)'

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you unset the variable using the **unsetenv** command, you remove the variable value in the application and in any new child processes you initiate from the application using the **exec** command. However, the variable is still set in the parent process.

See the **set** and **unset** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **unsetenv** remove the DISPLAY varible from the environment:

```
prompt> getenv DISPLAY
```

```
host:0
prompt> unsetenv DISPLAY
prompt> getenv DISPLAY
Error: can't read "::env(DISPLAY)": no such variable
        Use error_info for more info. (CMD-013)
```

SEE ALSO

catch(2)
exec(2)
printenv(2)
set(2)
unset(2)
setenv(2)
getenv(2)

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

SYNTAX

```
string unsuppress_message [messages]

list messages
```

ARGUMENTS

messages

A list of messages to enable.

DESCRIPTION

The **unsuppress_message** command provides a mechanism to re-enable the printing of messages which have been suppressed using **suppress_message**. You can suppress only informational and warning messages, so the **unsuppress_message** command is only useful for informational and warning messages. The result of **unsuppress_message** is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The **print_suppressed_messages** command displays currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*
prompt> unsuppress_message CMD-029
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
```

SEE ALSO

print_suppressed_messages(2)
suppress_message(2)

update_currents

Generates power and ground pin electrical currents that are needed for rail analysis.

SYNTAX

```
status update_currents
   [-static | -dynamic | -dynamic_vectorfree]
   [-reuse]
   [-reuse_power_file file_name]
   [-time window_list]
```

Data Types

```
string file_name
list window_list
```

ARGUMENTS

-static

Indicates that the power and ground pin currents are to be generated for static rail analysis.

-dynamic

Indicates that the power and ground pin currents are to be generated for dynamic rail analysis.

-dynamic_vectorfree

Indicates that the power and ground pins currents are to be generated for dynamic vector free rail analysis.

-reuse

Indicates to read in the previously generated analysis results. This option is used in conjunction with the analysis mode for which reuse needs to be performed.

-reuse_power_file

Indicates to read in the previously generated power report files from PrimeTime PX. To re-use power report file from static power analysis, the provided file should be a PrimeTime PX SAIF binary report file. This feature is only available in static rail analysis mode.

-time *window_list*

Specifies the time window for which power and ground currents are generated. By default, the power and ground pin currents are generated over the time window defined by the **read_vcd -time** option or the time window captured by the previously generated analysis results.

DESCRIPTION

The **update_currents** command generates the power and ground pin electrical currents that are needed to perform rail analysis. The command invokes the built-in PrimeTime PX power analysis technology to generate the necessary power and timing information for calculating currents of the design. Alternatively, you can choose to use the power and timing information that is generated from the previous run for calculating currents. The command returns 1 if successful, 0 otherwise.

The -static and -dynamic modes of this command are mutually exclusive. The static mode is used to prepare power and ground pin currents for static rail analysis, and the dynamic mode prepares power and ground pin currents for dynamic rail analysis. These modes are analagous to the *averaged* and *time-based* modes of the **update_power** command.

With the -static option enabled, the generated power and ground pin current for static analysis is a scalar value derived from the equation, I = P/V, where I stands for the current, P is the average power consumed by that pin, and V is the ideal voltage of the pin.

With the -dynamic option enabled, the generated power and ground pin current is represented as a time-varying waveform over the analysis window, and is derived from the instantaneous power consumed at the power and ground pin.

With the -dynamic_vectorfree option enabled, the generated power and ground pin current is represented as a time-varying waveform over the analysis window, and is derived from the dynamic vector free analysis of the design which typically does not require stimulus to be provided in the form of a VCD or a SAIF file.

With the -reuse option enabled, users can reuse the previously generated current data in another session. Note that this command needs to be used with the analysis mode for which previously generated data has to be reloaded or reused in the current session.

EXAMPLES

The following example shows a typical command sequence for generating rail currents for dynamic analysis. Note that the design has already been read in.

```
pr_shell> read_vcd MyVCDFile.vcd
pr_shell> update_currents -dynamic
```

The following example reuses the previously generated power, timing and current data in the dynamic analysis:

```
pr_shell> update_currents -dynamic -reuse
```

The following example performs the vector free dynamic rail analysis.

```
pr_shell> update_currents -dynamic_vectorfree
```

The following example reuses the results produced by the vector free dynamic rail analysis.

```
pr_shell> update_currents -dynamic_vectorfree -reuse
```

The following example reads in a full VCD event file and generates static currents based on a subset of the file:

```
pr_shell> read_vcd MyVCDFile.vcd
pr_shell> update_currents -static -time {1000 1010}
```

The following example specifies the rail database directory as ./my_rail_database. The **update_currents** command will generate power and ground pin currents and save them under the ./my_rail_database.

```
pr_shell> set rail_database ./my_rail_database
pr_shell> update_currents -static
```

The following example shows how to reuse the previously created power and ground pin currents.

```
pr_shell> set rail_database ./my_rail_database
pr_shell> update_currents -static -reuse
```

The following example illustrates how to reuse previously generated PrimeTime PX SAIF power report file:

```
pr_shell> update_currents -reuse -reuse_power_file "saif.rpt"
```

SEE ALSO

analyze_rail(2)
calculate_rail_voltage(2)
power_analysis_mode(3)
power_enable_analysis(3)
power_rail_output_file(3)
read_vcd(2)
set_power_analysis_options(2)
update_power(2)

update_em

Performs electromigration analysis based on the configuration options specified by using the **set_em_options** command.

SYNTAX

```
string update_em
```

ARGUMENTS

None

DESCRIPTION

The command performs electromigration analysis using the configuration options specified by the **set_em_options** command. When the analysis run is completed, the tool displays the electromigration map. You can also view the errors in the error browser.

If you perform analysis with the electromigration map or error browser opened in the GUI, the tool automatically updates the electromigration map with the latest data.

EXAMPLES

```
pr_shell> update_em
1
```

SEE ALSO

```
set_em_options(2)
report_em_options(2)
report_em_rules(2)
report_em_violations(2)
```

which

Locates a file and displays its pathname.

SYNTAX

`string` **`which`** *`filename_list`*

`list` *`filename_list`*

ARGUMENTS

filename_list

List of files to locate.

DESCRIPTION

Displays the location of the specified files. This command uses the search_path to find the location of the files. This command can be a useful prelude to read_db or link_design, because it shows how these commands expand filenames. The **which** command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

EXAMPLES

The following examples are based on the following search_path.

prompt> **set search_path "/u/foo /u/foo/test"**

The following command searches for the file name foo1 in the search_path.

```
prompt> which foo1
/u/foo/foo1
```

The following command searches for files foo2, foo3.

```
prompt> which {foo2 foo3}
/u/foo/test/foo2 /u/foo/test/foo3
```

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db
/u/foo/test/designs/sub_design.db
```

SEE ALSO

link_design(2)
read_db(2)
search_path(3)

write_activity_waveforms

Creates activity waveforms from the Value Change Dump (VCD).

SYNTAX

```
integer write_activity_waveforms
   -interval sampling_interval
   [-coverage]
   [-hierarchical_levels level]
   [-exclude_signals signal_list]
   [-exclude_cells module_list]
   [-output file_name]
   [-format fsdb | out]
   [-time time_list]
   -vcd filename
   [-verbose]
   [-peak_type min | max]
   [-peak_window window_size]
   [-strip_path path]
```

Data Types

sampling_interval	float
level	integer
signal_list	list
module_list	list
file_name	string
time_list	list
filename	string
window_size	string
path	string

ARGUMENTS

-interval *sampling_interval*

Specify the sampling interval that is used for activity waveform. during each interval, activity is aggregated.

-coverage

Define activity to be the percentage of nets in a block that toggle at least one time during an interval. By default, the definition of activity is the number of toggles in the block for the interval, divided by number of nets and the interval period.

-hierarchical_levels *level*

Create power waveforms for hierarchical cells only to the specified level from the top. The default is to create waveforms for only the top level of the design.

-exclude_signals *signal_list*

This option allows you to exclude some signal names from consideration. The list of signals is a list of signal names. No path should be given with the signal names; all signals with the specified names are ignored from the VCD. This option is useful for excluding signals, such as clock signals from analysis.

-exclude_cells *module_list*

This option allows you to exclude some module names from consideration. The list of modules is a list of module names. The names can either exclude a path to the module (such as C instead of TOP/A/B/C), or the full path can be included. If no path is given, all modules with the specified names are ignored from the VCD. The given module names can be globs, such as A*, which would exclude any module starting with 'A'. When a module is excluded, it is not included in the waveforms or the text reports (see the **-verbose** option). Also, the signals in the excluded modules are not considered in the hierarchy above the excluded module. As an exception to this, when a signal is defined in TOP/A/B and also defined in TOP/A/B/C, the signal is still counted as being in B if module C is excluded. The **-exclude_cells** option is useful for excluding macro modules or unsynthesized modules.

The given list of modules needs to be a list of module names in the VCD. It should not be a collection of cells from a design.

-output *file_name*

Specify the prefix of the file in which the waveform data is to be written.

-format fsdb | out

Specifies the format for the output file. Valid formats are **fsdb** and **out**. The default is **fsdb**.

-time *time_list*

Specifies the time windows for the analysis. You might want to only analyze and produce waveforms for a portion of the time in the VCD. The **-time** option accepts a list of numbers as its argument. The argument must have an even number of elements. Each consecutive pair of numbers is a time window; the first number is the start of the window, the second is the end of the window. Activity analysis is only performed on the given time windows.

The given time values are in nanoseconds. The first number in the list can use the special value -1 to represent the beginning of the VCD. The last number in the list can use the special value -1 to represent the end of the VCD.

-vcd *filename*

Specifies the file name of the VCD to be analyzed. This option is required.

-verbose

After analysis, a report is generated, similar to the **report_activity_waveforms** command.

-peak_type min | max

Specifies whether the command is to search for an interval with maximum or minimum activity (or coverage, if the **-coverage** option is also specified). By default, the command searches for an interval with maximum activity (or coverage).

-peak_window *window_size*

Specify the window size to be used when searching for the peak activity period. By default, the peak_window is the same as the interval. PrimeTime PX searches for the peak activity period for each hierarchical block during activity file analysis. The width of the period is the peak_window. For example, the interval might be set to 10 ns, while the peak_window could be 500 ns. In this case, PrimeTime PX looks for activity in a 500 ns window

when deciding when the peak activity occurs.

Due to implementation limitations, in the example, the 500 ns window starts and interval boundaries are on. Therefore, the value specified for the **-peak_window** option must be a multiple of the interval value.

-strip_path ***path***

Specifies the path to the module in the VCD of interest. All modules outside the hierarchy under the specified path are excluded. The names in the waveform file are shortened.

DESCRIPTION

Plots the average toggle rate for modules in the VCD over time. The toggle rate during an interval for a module is the average toggle rate for signals in the module for the interval.

No design or library needs to be loaded to use this command; only a VCD is needed.

Activity waveforms can be useful for qualifying the activity vectors. You can review the waveforms to determine if the testbench simulated as expected, and whether the vectors have covered enough of the design to be useful as inputs to power analysis.

The total VCD simulation time is partitioned into intervals, for which the average activity is calculated based on the following formula:

```
                          # of toggles on all the signals for the interval
Average toggle rate =  ----------------------------------------------------
                         (# of signals) * (length of interval in nanoseconds)
```

You can also view the activity coverage over time to get a sense of how many of the signals are being exercised. A signal is "covered" if it has at least one toggle within the interval, so the coverage per interval is calculated as follows:

```
             # of signals with at least one toggle
Coverage = ---------------------------------------- * 100%
                     # of signals
```

The output format for the waveforms is either the default of .fsdb or the .out format. View either format with a waveform viewer, such as nWave.

EXAMPLES

This example outputs the file#.out text file with waveform information in the .out format:

```
pt_shell> write_activity_waveforms -output file1 -vcd my_vcd.vcd -interval 10
```

This example analyzes the time window from 100 to 300 and then from 1000 to the end of the vcd:

```
pt_shell> write_activity_waveforms -output file2 \
            -vcd my_vcd.vcd -interval 10 -time {100 300 1000 -1}
```

Use the **-exclude_cells** option to exclude some modules from the waveforms and to exclude nets in the modules from being included in the totals for other modules higher in the hierarchy. For example,

```
pt_shell> write_activity_waveforms -output file3 -vcd my_vcd.vcd \
             -interval 10 -exclude_cells {assertion_module* other_module}
```

SEE ALSO

read_vcd(2)
report_activity_waveforms(2)

write_app_var

Writes a script to set the current variable values.

SYNTAX

```
string write_app_var
   -output file
   [-all | -only_changed_vars]
   [pattern]
```

Data Types

```
file        string
pattern     string
```

ARGUMENTS

-output *file*

Specifies the file to which to write the script.

-all

Writes the default values in addition to the current values of the variables.

-only_changed_vars

Writes only the changed variables. This is the default when no options are specified.

pattern

Writes the variables that match the specified *pattern*. The default is "*".

DESCRIPTION

The **write_app_var** command generates a Tcl script to set all application variables to their current values. By default, variables set to their default values are not included in the script. You can force the default values to be included by specifying the **-all** option.

EXAMPLES

The following is an example of the **write_app_var** command:

```
prompt> write_app_var -output sh_settings.tcl sh*
```

SEE ALSO

```
get_app_var(2)
report_app_var(2)
set_app_var(2)
```

write_arrival_annotations

Writes arrival and slew annotations for ILMs or contexts of the given list of instances or for all top level instances as a script of commands.

SYNTAX

```
status write_arrival_annotations
   [-instances instance_list]
   [-context]
   [-design]
```

Data Types

```
instance_list     list
```

ARGUMENTS

-instances *instance_list*

Writes arrival annotations for the set of pins affecting the given list of instances. The tool looks for appropriate files that contain the list of aggressor driver pins inside the directories meant for these instances, and creates corresponding scripts.

-context

Specifies that the annotations are meant for the context of the block.

-design

Specifies that the annotations are meant for the entire design.

DESCRIPTION

This command writes arrival annotations to be used at block-level or chip-level hierarchical signal integrity analysis. This command is affected by the **pt_ilm_dir** environment variable. The tool looks for directories meant for the list of instances (or all top level instances) at the location given by the **pt_ilm_dir** environment variable. Inside the directory, it looks for wrapper.txt file or ilm.txt file depending on whether the **-context** option is used or not. Then it creates a new file called wrapper.pt.gz or ilm.pt.gz that contains the script needed to annotate the arrivals and slews at block or chip level.

If the annotations are for the chip level and the ILM is being generated for the full design (this happens if the blocks are shielded from each other and the signal integrity context of the instance is not generated), then use the **-design** option. Then, PrimeTime looks for a directory with the name of the block design and looks for ilm.txt

file inside the directory.

EXAMPLES

The following example writes the annotations for contexts of all top level blocks:

```
pt_shell> write_arrival_annotations -context
```

The following example writes annotations for the ILM of instance I1:

```
pt_shell> write_arrival_annotations -instances {I1}
```

SEE ALSO

create_ilm(2)
create_si_context(2)
pt_ilm_dir(3)

write_changes

Outputs netlist changes that have been recorded during this session.

SYNTAX

```
status write_changes
   [-format ptsh | text | dctcl | icctcl | icc2tcl | eco]
   [-output file_name]
   [-reset]
```

Data Types

```
file_name     string
```

ARGUMENTS

-format ptsh | text | dctcl | icctcl | icc2tcl | eco

Specifies one of the following output formats:

- **ptsh** (default) - PrimeTime script
- **text** - Text
- **dctcl** - Tcl script for Design Compiler; compatible with Design Compiler version Z-2007.03 and later releases
- **icctcl** - Tcl script for IC Compiler; compatible with IC Compiler version Z-2008.12, IC Compiler II J-2014.12-SP1-1 and later releases
- **icc2tcl** - Tcl script for IC Compiler II
- **eco** - Binary script for the PrimeTime **read_eco_changes** command

-output *file_name*

Writes the change list to the specified file. If you do not use this option, the change list goes to the standard output.

-reset

Clears the netlist change list so that any history of previous netlist editing commands is no longer maintained. This options does not undo the effect of the previous netlist editing commands; it simply clears the netlist change list.

DESCRIPTION

The **write_changes** command outputs the netlist changes that were made to the current design after it was linked. It is typically formatted as a script. By default, the output goes to the standard output. To write the output to a file, use the **-output** option.

The following commands create entries in the change list:

```
connect_net
create_cell
create_net
disconnect_net
insert_buffer
remove_buffer
remove_cell
remove_coupling_separation
remove_net
rename_cell
rename_net
set_coupling_separation
size_cell
swap_cell
```

As a script, the output of the **write_changes** command is a series of scoping commands (**current_instance**) and netlist editing commands. The text format is basically one line per atomic editing operation, in order, describing in words the operation, the instance in which it occurred, and the objects involved.

The change list is not a verbatim log of what you entered. In the following cases, the change list might be different from the command history:

- When the objects were specified at a higher level of the hierarchy (for example, creating a cell within u1/u2), the change list would scope to u1/u2 using the **current_instance** command, then create the cell.
- Some object arguments are made explicit by wrapping them within a command which creates a collection, such as the **get_pins** command.
- Often, multiple arguments to commands, such as the **create_cell** command, are converted into separate commands in the change list.
- Sometimes some arguments are dropped from the change list, such as when superfluous arguments are given to the **remove_cell** command.

The **write_changes** command attempts to merge together some redundant operations to reduce and simplify the change list. For example, if a buffer is inserted then removed, this simplifies to a null operation. If a cell is resized multiple times, this is simplified to a single resize operation. Note: Simplification does not occur for the text format.

The following commands are not written to the Design Compiler Tcl scripts (the **dctcl** format):

```
remove_coupling_separation
rename_cell
rename_net
set_coupling_separation
swap_cell
```

The following commands are not written to the IC Compiler Tcl scripts (the **icctcl** format):

```
create_net - If the net name contains wildcard character or hierarchy separator.
remove_coupling_separation
rename_cell
rename_net
set_coupling_separation
```

```
swap_cell
```

IC Compiler does not support the **create_cell -exact** command. If the **-exact** option is used, the **create_cell** command is written out without the **-exact** flag when the **icctcl** format is selected.

Physical ECO

After PrimeTime ADV performs physical ECO by using physical data, generate the change list file by using the **write_changes** command. The change list contains the placement location of new buffers. If you use the **-format icctcl** option, the change list is compatible with IC Compiler versions 2013.03-SP2 and later.

After you run physical ECO, the tool disables the **ptsh** format. Because PrimeTime ADV keeps track of physical changes internally, do not use the **-reset** option after the physical data is loaded into memory. Binary format **eco** is used to replay ECO changes back to PrimeTime. The **eco** format changes can be read by **read_eco_changes** command.

Multiply Instantiated Modules (MIM)

When the **eco_enable_mim** variable is true, the **write_changes** command writes one change list for the associated MIM so that the change list can be directly implemented by the place-and-route tool such as IC Compiler.

Suppose a CPU module is instantiated four times in TOP module as CPU1, CPU2, CPU3, and CPU4. After fixing is completed with MIM enabled, the following command is used to write change lists:

```
pt_shell> write_changes -format icctcl -output pteco.tcl
```

This creates the following two change lists in the current directory:

```
TOP_pteco.tcl      # A change list for TOP module
CPU_pteco.tcl      # A change list for CPU module
```

The specified output file name is used as postfix after the name of the MIM. Also the file contains the instance names of the changes to avoid any confusion.

EXAMPLES

The following shows a set of netlist editing commands and the change list created by the edits. Notice how the creation of two cells in a single command is converted to two entries in the change list.

```
pt_shell> create_cell i1/u1 class/AN2
Uniquifying 'i1' (inter) as 'inter_0'.
Information: Created cell 'u1' in 'middle/i1'. (NED-014)
1
pt_shell> size_cell i1/u1 class/AN2P
Information: Sized 'i1/u1' with 'class/AN2P'. (NED-045)
1
pt_shell> create_net i1/net1
Information: Created net 'net1' in 'middle/i1'. (NED-016)
1
pt_shell> connect_net i1/net1 i1/u1/A
Information: Connected 'i1/u1/A' to 'net1'. (NED-018)
1
pt_shell> create_cell {u1 u2} class/AN2
Information: Created cell 'u1' in design 'middle'. (NED-014)
Information: Created cell 'u2' in design 'middle'. (NED-014)
1
pt_shell> write_changes -format ptsh
```

```
#######################################
# Change list, formatted for PrimeTime
#######################################
current_instance
current_instance i1
create_cell {u1} {class/AN2}
size_cell {u1} {class/AN2P}
create_net {net1}
connect_net {net1} [get_pins {u1/A}]
current_instance
create_cell {u1} {class/AN2}
create_cell {u2} {class/AN2}
1
pt_shell> write_changes -format text
Change list, formatted as text:


 1. create_cell in i1: new cell 'u1' lib_cell 'class/AN2'

 2. size_cell in i1: 'u1' sized to 'class/AN2P'

 3. create_net in i1: new net 'net1'

 4. connect_net in i1: pin 'u1/A' to net 'net1'

 5. create_cell in <top>: new cell 'u1' lib_cell 'class/AN2'

 6. create_cell in <top>: new cell 'u2' lib_cell 'class/AN2'



1
```

The following example shows how the change list specifies the location information for IC Compiler.

```
pt_shell> write_changes -format icctcl
#######################################
# Change list, formatted for IC Compiler
#######################################
current_instance
add_buffer_on_route net1 -user_specified_buffers {U1 BUF1X 102.0 202.0 0}
set_cell_location -coordinates {100.0 200.0} -orientation NF \
                  [get_cells U1 -hierarchical]
insert_buffer U200/Z BUF2X -location {500.0 700.0}
size_cell U100 BUF4X
```

SEE ALSO

```
connect_net(2)
create_cell(2)
create_net(2)
current_instance(2)
disconnect_net(2)
fix_eco_drc(2)
fix_eco_power(2)
fix_eco_timing(2)
insert_buffer(2)
remove_buffer(2)
remove_cell(2)
remove_coupling_separation(2)
```

```
remove_net(2)
rename_cell(2)
rename_net(2)
set_coupling_separation(2)
set_eco_options(2)
size_cell(2)
swap_cell(2)
```

write_context

Writes the timing context information for specified objects as a Tcl script or in binary context format.

SYNTAX

```
status write_context
   [-clock]
   [-compress]
   [-output file_directory_name]
   [-boundary_only]
   [-format ptsh | sdc | dctcl | gbc]
   [-nosplit]
   [cell_block_list]
```

Data Types

file_directory_name	string
cell_block_list	list

ARGUMENTS

-clock

Writes only clock information of script constraints. By default, all context information is written.

-compress

Generates constraint scripts in compressed format.

-output *file_directory_name*

Specifies the name of the file or directory in which to write the context. A script for a single object is written to a file. Scripts for multiple objects are written to a directory. Binary context information (-format gbc) is written to a directory.

-boundary_only

Generates a script that includes only the boundary I/O context.

-format ptsh | sdc | dctcl | gbc

Specifies the output format for the script. The allowed values are:

ptsh	PrimeTime script format
sdc	Synopsys Design Constraints script format
dctcl	Design Compiler script format
gbc	binary context format

-nosplit

Disables line splitting, resulting in long lines. This option is useful for comparing different versions of generated scripts and for postprocessing the generated script.

cell_block_list

Specifies a list of instances or blocks for which the context is written out. If no object is given, the default is to write out all objects that have been characterized with **characterize_context** command.

DESCRIPTION

This command writes the timing context of the specified instances or blocks as a PrimeTime, SDC, or Design Compiler script; or in binary context format. The context must first be captured using the **characterize_context** command. You can use the generated context information to set the timing constraints for a block design for synthesis or logic optimization in another tool, or for performing hierarchical timing analysis in the PrimeTime tool.

You can use the command options to specify the type of script to generate and the name of the script file, or specify a directory in which to write out the context information in binary format. If you run the **write_context** command without options, the command writes out all context information to standard output in PrimeTime script format.

To use context information written in script format, source the script using the appropriate tool. To use the context information written in binary format, use the **read_context** command.

EXAMPLES

The following example writes to standard output the context information for instances A1 and A2, in Design Compiler format.

```
pt_shell> write_context -format dctcl {A1 A2} -output myDC_dir
```

The following command writes all context information in PrimeTime script format to a file named A1.ptsh in the current directory.

```
pt_shell> write_context -output A1
```

The following example writes all context information for instance A1 in binary context format using the directory named A1BC.

```
pt_shell> write_context -format gbc -output A1BC A1
```

SEE ALSO

```
characterize_context(2)
read_context(2)
report_context(2)
write_physical_annotations(2)
```

write_interface_timing

Generates an interface timing ASCII report for a gate-level netlist, an interface logic model (ILM), or an extracted timing model (ETM) design.

SYNTAX

```
int write_interface_timing
   file_name
   [-ignore_ports port_list]
   [-significant_digits digits]
   [-nosplit]
   [-timing_type slack | arc]
   [-verbose]
   [-include incl_list]
   [-latch_level level]
```

Data Types

```
file_name     string
port_list     list
digits        int
incl_list     list
level         int
```

ARGUMENTS

file_name

Specifies the name of the file to which the interface timing information is to be written.

-ignore_ports *port_list*

Specifies a list of ports that are to be excluded from the timing file. By default, all ports are included.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the **report_timing** command. Allowed values are 0-13.

The default is the maximum of six and the value of the **report_default_significant_digits** variable. Use this option if you want to override the default. Fixed-precision floating-point arithmetic is used for delay calculation; thus, the actual precision might depend on the platform.

-nosplit

Prevents line-splitting. This is most useful for doing diffs on previous scripts or for postprocessing the script.

-timing_type slack | arc

Selects the type of timing data section to be reported.

- **slack** (default) - Reports the slack associated with interface timing relationships.
- **arc** - Reports the timing arc values for interface timing relationships.

Slack values are a function of the external environment (clocks, arrival times), but arc values are not.

If you are validating an ETM that gave a MEXT-54 warning when generated, do not use arc values. When the MEXT-54 warning is given, some of the ETM arcs have been adjusted by a multiple of the clock period and does not match the arc values of the original netlist.

-verbose

Shows details of the progress of the **write_interface_timing** command and provides data on CPU time and memory usage. For longer runs, it shows a time estimate of completion.

-include *incl_list*

Selects the type of data to analyze. You can specify the following values for the *incl_list*:

- **timing** (default) - Reports the slack/arc, capacitance, transition time, and design rules sections.
- **noise** - Reports the noise sections noise_detection and noise_calculation.

-latch_level *level*

Specifies the number of levels of latch borrowing that are to occur at the interface of a design. The default is 12.

DESCRIPTION

This command generates an interface timing ASCII report for a gate-level netlist, an interface logic model (ILM), or an extracted timing model (ETM) design. The report is a file containing interface timing, slew, capacitance, design rule, and noise information for a gate-level netlist, an ILM, or an ETM design. The command performs an implicit **update_timing** and **update_noise**, if necessary, for the selected data sections. A return code of *1* indicates that the command ran to completion; a return code of *0* indicates an error.

The output report file contains the following sections:

Slack - Shows worst-case slacks for combinational paths from an input port to an output port, for input-to-register paths from an input port to each clock used to clock the input port, and for output-to-register paths from each clock used to clock an output port to the output port.

For each port-to-port, register-to-port, and port-to-register timing relationship, it reports slacks for the following possible arc types: min_seq_delay, max_seq_delay, min_combo_delay, max_combo_delay, setup, hold, recovery, removal, clock_gating_setup, and clock_gating_hold.

Arc - Shows worst-case setup/hold/delay arc values for combinational paths from an input port to an output port, for input-to-register paths from an input port to each clock used to clock the input port, and for output-to-register paths from each clock used to clock an output port to the output port.

For each port-to-port, register-to-port, and port-to-register timing relationship, it reports arc values for the following possible arc types: min_seq_delay, max_seq_delay, min_combo_delay, max_combo_delay, setup, hold, recovery, removal, clock_gating_setup, and clock_gating_hold.

Transition Time - Shows actual transition times on all ports for the four delay types: min_fall, min_rise, max_fall, and max_rise.

Input Capacitance - Shows actual total (lumped) or effective capacitances on boundary nets connected to each port.

Design Rule - Shows design rule information for all ports: max_capacitance, min_capacitance, max_transition, max_fanout, and fanout_load.

Noise Detection - Shows noise slack information for all input ports for the four noise regions: below low, above low, below high, above high.

Noise Calculation - Shows driver steady-state resistance for all output ports for the four noise regions: below low, above low, below high, above high.

To compare two report files generated by the **write_interface_timing** command, use the **compare_interface_timing** command.

EXAMPLES

The following example writes timing information for the current design to the model.rpt file.

```
pt_shell> write_interface_timing -timing_type arc model.rpt
```

The following example writes noise information for the current design to the noise_net.rpt file.

```
pt_shell> write_interface_timing -include noise noise_net.rpt
```

SEE ALSO

compare_interface_timing(2)
update_noise(2)
update_timing(2)
report_default_significant_digits(3)

write_parasitics

Writes out annotated parasitics information for the current design.

SYNTAX

```
status write_parasitics
   -format SPEF | GPD
   -no_name_mapping
   -nets net_list
   file_name
```

Data Types

net_list	list
file_name	string

ARGUMENTS

-format SPEF | GPD

Specifies the format of the output parasitics file. Currently, the only allowed values are Standard Parasitic Exchange Format (**SPEF**) and Galaxy Parasitics Database (**GPD**).

-no_name_mapping

This option is only for SPEF. When specified for SPEF, it means that name maps should not be used for generating SPEF file. By default, a name map section is always generated.

-nets *net_list*

Specifies the list of nets to output parasitics in the parasitics file. This option is supported for both SPEF and GPD.

file_name

Specifies the name of the output parasitics file.

DESCRIPTION

The **write_parasitics** command writes all parasitics annotated on the current design to the file specified in the *file_name* argument. This command supports the SPEF and GPD formats, and a *PrimeTime SI* license is required for GPD.

Binary parasitics are useful for analysis iterations. You can read parasitics in one format (for example, SPEF), then use the **write_parasitics** command to output the parasitics in GPD format, which loads much faster than ASCII formats and is much more compact. The **write_parasitics** command can write detailed RC, PI model, and lumped-capacitance networks.

SPEF writing can be useful for debugging if you read parasitics into PrimeTime through GPD.

EXAMPLES

The following example reads the top.spef parasitics file and then writes it out in GPD format:

```
pt_shell> read_parasitics top.spef
****************************************
Report : read_parasitics /u/designs/top.spef
Design : TOP
****************************************
        0 error(s)
        Format is SPEF
        Number of annotated nets          :        66
        Number of annotated capacitances  :      1390
        Number of annotated resistances   :      1333
        Number of annotated RC pi models  :         0
        Number of annotated Elmore delays :         0
1
pt_shell> write_parasitics -format GPD top.gpd
1
pt_shell> write_parasitics -format SPEF top.spef
1
```

SEE ALSO

read_parasitics(2)

write_physical_annotations

Writes annotated delays and parasitics for a hierarchical cell.

SYNTAX

```
string write_physical_annotations
   [-sdf sdf_file]
   [-version sdf_version]
   [-nets_only]
   [-cells_only]
   [-boundary_nets]
   [-parasitics parasitics_file]
   [-append script_file]
   [-format ptsh | dcsh | dctcl]
   cell
```

Data Types

```
sdf_file            string
sdf_version         string
parasitics_file     string
script_file         string
cell                string
```

ARGUMENTS

-sdf *sdf_file*

Indicates that annotated delays are to be written in Standard Delay Format (SDF) to the specified *sdf_file*. You must specify either **-sdf** or **-parasitics**, but you cannot specify both. If you specify either the **-version**, **-nets_only**, or **-cells_only** option, you must also specify the **-sdf** option.

-version *sdf_version*

Specifies the version of the SDF to use. Allowed values are **1.0** and **2.1** (the default). If you specify the **-version** option, you must also specify the **-sdf** option.

-nets_only

Indicates that only delays annotated on nets be written out as SDF. By default, all annotated delays are written out. If you specify the **-nets_only** option, you must also specify the **-sdf** option.

-cells_only

Indicates that only delays and timing checks annotated on cells be written out as SDF. By default, all annotated delays are written out. If you specify the **-cells_only** option, you must also specify the **-sdf** option.

-boundary_nets

Indicates that parasitics annotated on boundary nets of the specified cell must also be written out to the parasitics file. By default, only the annotated parasitics of internal nets are written out. Do not use this option if you issue this command in conjunction with the **characterize_context** command to avoid overwriting characterized capacitance on boundary nets. If you specify the **-boundary_nets** option, you must also specify the **-parasitics** option.

-parasitics ***parasitics_file***

Indicates that annotated capacitance and resistance on nets are to be written to the specified *parasitics_file* as a script. Annotated capacitance is written out as a **set_load** command for for pt_shell and dc_shell. Annotated resistance is written out as a **set_resistance** command. You must specify either the **-sdf** or **-parasitics** option, but you cannot specify both options. If you specify the **-boundary_nets** option, you must also specify the **-parasitics** option.

-append ***script_file***

Appends commands to import the generated files in the given script file. The SDF file is read using the **read_sdf** command in pt_shell and the **read_timing** command in dc_shell. The parasitics file is included in the script file.

-format ptsh | dcsh | dctcl

Specifies the format in which to output the commands in the script file and the parasitics file. Allowed values are **ptsh** (the default), **dcsh**, or **dctcl**.

cell

Specifies the name of the cell for which to export the annotations. The cell must be hierarchical.

DESCRIPTION

The **write_physical_annotations** command writes physical information for a hierarchical block in the design. The exported information includes annotated net and cell delays using the Standard Delay Format (SDF). Annotated parasitics can also be exported as a series of pt_shell or dc_shell commands.

The **write_physical_annotations** command is most commonly used in conjunction with the **characterize_context** and **write_context** commands to export timing and physical information for a block from the chip-level environment to module level tools. A typical sequence of commands is to first characterize the timing environment of a block using the **characterize_context** command and write this information as a dc_shell script using the **write_context** command. The **write_physical_annotations** command is then used to export annotated delays and parasitics for internal nets of the block.

To avoid overwriting characterized capacitance on boundary nets, do not specify the **-boundary_nets** option. You can use the **-append** option to append commands to read the SDF and include the parasitics script at the end of the script file generated by the **write_context** command.

EXAMPLES

The following command writes the annotated delays on nets for the hierarchical cell 'I1':

```
pt_shell> write_physical_annotations \
            -sdf I1.sdf -nets_only I1
```

The following command is used in conjunction with the **characterize_context** and **write_context** commands to export the timing and physical environment for cell 'I2' as a dc_shell script. The command uses the **append** option to augment the script generated by the write_context command with commands to import the generated delay and parasitic files.

```
pt_shell> characterize_context I2

pt_shell> write_context -format dcsh \
          -out I2.dcsh I2

pt_shell> write_physical_annotations \
          -sdf I2.sdf -parasitics I2.rc \
          -format dcsh -append I2.dcsh \
          I2
```

SEE ALSO

```
characterize_context(2)
read_sdf(2)
remove_context(2)
report_context(2)
set_load(2)
set_resistance(2)
write_context(2)
write_sdf(2)
```

write_rail_data

Writes rail data from the current session into a text file that can be used to set the relevant rail data in another session.

SYNTAX

```
status write_rail_data
   -type type_name
   -supply_net supply_nets
   -threshold threshold
   -limit count
   -cell cells
   file_name
```

Data Types

```
string file_name
string type_name
float threshold
integer count
cell_collection cells
supply_net_collection supply_nets
```

ARGUMENTS

file_name

Specifies the name of the file to which the rail data is to be written.

-type *type_name*

Specifies what type of rail data to be written. The supported data types are instance_power, pg_pin_power, voltage_drop_or_rise, effective_voltage_drop_or_rise, minimum_path_resistance and pg_pin_peak_current.

-threshold *threshold*

Specifies a threshold to filter out the rail data to be written. When specified, rail data which is smaller than the specified threshold will be filtered out. Note that this option does not applicable to the minimum_path_resistance data type.

-limit *count*

Specifies the number of elements to be written. The default is 0, meaning all data is to be reported.

-cell *cell_instances*

Specifies the cell instances to be reported. Note that this option is not applicable to the voltage_drop_or_rise and effective_voltage_drop_or_rise data types.

-supply_net ***supply_net_objects***

Specifies the supply net objects to be reported. Note that this option is not applicable to the instance_power data type.

DESCRIPTION

The command writes a file containing rail data from the current session; the rail analysis data written to the file is sorted in a descending order. When writing instance power values, you can import the output file into a later session with the **set_rail_power** -import *file* command.

The rail data written to the file can be limited to a specific type by using the -type option. The supported types are:

```
instance_power: Cell instance power values
pg_pin_power: Cell instance PG pin power values
voltage_drop_or_rise: Dynamic voltage drop for PG pins
effective_voltage_drop_or_rise: Effective voltage drop between PG pins
minimum_path_resistance: Minimum resistance values with cell instances.
pg_pin_peak_current: Peak current values for PG pins
```

The following is the format of the output file for each type:

```
instance_power: CELL full_path_cell_instance_name power
pg_pin_power: PG_PIN full_path_cell_instance_name pg_pin_name power
voltage_drop_or_rise: full_path_cell_instance_name/pg_pin_name voltage
effective_voltage_drop_or_rise: full_path_cell_instance_name pg_pin_name,pg_pin_name voltage
minimum_path_resistance: full_path_cell_instance_name/pg_pin_name resistance
pg_pin_peak_current: full_path_cell_instance_name/pg_pin_name current
```

The power unit in the report file is Watt, the current unit is Amp, and the voltage unit is Volt.

The rail data written to the report file can be filtered by using the -threshold *threshold* option. If the rail data is greater than the specified threshold value, that rail data will be written to the file. The supported types are: instance_power, pg_pin_power, voltage_drop_or_rise, effective_voltage_drop_or_rise and pg_pin_peak_current.

The number of elements written to the file can be specified using -limit *number_of_elements* option. Only the specified number of elements are included in the file in a descending order of the value. If the -limit option is not specified or is set to 0, the tool writes all elements to the output file. The -elements option supports all data types.

Use the -cell *cell_instances* option to reports details of the specified cell instances. The supported types are: instance_power, pg_pin_power, minimum_path_resistance and pg_pin_peak_current.

Use the -supply_net *supply_net_objects* option to write rail data for the specified supply nets. The supported types are: pg_pin_power, voltage_drop_or_rise, effective_voltage_drop_or_rise, minimum_path_resistance and pg_pin_peak_current.

EXAMPLES

The following example writes a file containing cell instance power which is later read in by the **set_rail_power** command.

```
pr_shell> write_rail_data -type instance_power set_power_file
```

```
Information: total 188 of cells written to file "set_power_file" with total power: 0.0001 (W)
pr_shell> set_rail_power -import set_power_file
Information: 188 of power values assigned to cells from the file set_power_file
```

The following example writes a file containing top 5 peak pg pin current values.

```
pr_shell> write_rail_data -type pg_pin_peak_current -limit 5 peak.rpt
1
pr_shell> sh cat peak.rpt
FI2/vss 4.81004e-06
FI6/vss 4.80959e-06
FI3/vss 4.79702e-06
FI4/vss 4.79684e-06
FI1/vss 4.79594e-06
```

The following example writes a file containing minimum path resistances.

```
pr_shell> write_rail_data -type minimum_path_resistance minres.rpt
1
pr_shell> sh cat minres.rpt
FEED0_14/vbp 518.668
INV61/vdd 158.268
ARCR0_1/vbp 145.582
FEED0_5/vdd 156.228
INV11/vbp 518.669
...
```

The following example writes a file containing minimum path resistances for the cell instance INV61.

```
pr_shell> write_rail_data -type minimum_path_resistance -cell INV61 minres.rpt
1
pr_shell> sh cat minres.rpt
PG Pin : INV61/vss
    M1 (447.060, 135.990) (447.180, 136.350)       158.268
    M1 (441.420, 135.990) (447.060, 136.350)
    M1 (441.300, 135.990) (441.420, 136.350)       156.228
    M1 (439.620, 135.990) (441.300, 136.350)
...
    M1 (409.260, 135.990) (409.380, 136.350)       144.881
    M1 (407.940, 135.990) (409.260, 136.350)
    M1 (407.820, 135.990) (407.940, 136.350)       144.37
    M1 (0.360, 135.990) (407.820, 136.350)
TAP M1 (0.000, 135.990) (0.360, 136.350)
...
```

SEE ALSO

set_rail_power(2)

write_rlib_template

Generates a template for the rail liberty file.

SYNTAX

```
status write_rlib_template
   [-db db_name]
   [-cell cell_name]
   [-output template_file_name]
```

Data Types

```
string db_name
string cell_name
string template_file_name
```

ARGUMENTS

-db *db_name*

Specifies the name of the Synopsys .db file where the tool reads the information for generating the rail liberty template.

-cell *cell_name*

Specifies the name of the hard macro cell.

-output *template_file_name*

Specifies the name of the rail liberty file to be generated.

DESCRIPTION

This command generates a rail liberty template file in which describes the electric model of a hard macro cell, depending on the specified Synopsys .db file. The electric model includes the PVT corners, the trigger conditions of a power event, and so on. If no .db file is specified, the default values are used.

EXAMPLES

The following example generates a rail liberty template file for the hard macro cell MyCell.

```
pr_shell> write_rlib_template -db MyDB -cell MyCell -output MyCell.rlib
1
```

SEE ALSO

hard_macro_rail_liberty_file(2)

write_saif

Writes a backward Switching Activity Interchange Format (SAIF) file.

SYNTAX

```
int write_saif
   file_name
   [-cells cell_list]
   [-derate_glitch value]
   [-rtl]
   [-propagated]
   [-exclude_sdpd]
   [-no_hierarchy]
   [-compress compression]
```

Data Types

```
file_name          string
cell_list          list
value              float
```

ARGUMENTS

file_name

Specifies the name of the output SAIF file.

-cells *cell_list*

Specifies an optional list of hierarchical instances for which the SAIF file is generated. If this argument is not specified, the SAIF file is generated for the current instance.

-derate_glitch *value*

Specifies a derating factor value to be used for converting part of glitches into inertial glitches. If this argument is not specified, a default derating factor of 0.5 is used.

-rtl

Specifies that the generated SAIF file contains the switching activity for the synthesis invariant objects. These are the design ports, hierarchical cell pins, and outputs of sequential and tri-state cells. Note that you should not use the **-rtl_direct** option of the **read_saif** command when reading SAIF files generated by the **write_saif** command.

-propagated

Propagates the user-annotated switching activity to estimate the switching activity of unannotated objects, and generates a SAIF file containing both the annotated and estimated switching activity. When this option is not

specified, the **write_saif** command generates a SAIF file containing only the user-annotated switching activity.

-exclude_sdpd

Inhibits the state-dependent and path-dependent (SDPD) switching activity generation. If you do not specify this option, the **write_saif** command outputs SDPD information by default. When the **write_saif** command is called with the **-propagated** option, the estimated SDPD information is included in the SAIF file.

-no_hierarchy

Specifies that the SAIF file contains switching activity information for only the top-level design objects. When this option is not specified, the **write_saif** command generates a SAIF file containing switching activity information for all the objects in the hierarchy.

-compress

Specify that the output saif file should be compressed. The only valid value for compression is gzip.

DESCRIPTION

The **write_saif** command generates a backward SAIF file containing the user-annotated and propagated simple and SDPD switching activity in the current design.

The top-level instance name in the generated SAIF file is the current instance name. You must specify this name as the instance name when reading the SAIF file with the **read_saif** command.

By default, the saif file is written as simple text. Optionally, the file can be compressed using the -compress option.In some cases, a file extension is appended to file_name if it is omitted. For example, with gzip formatted files, if file_name does not end with ".gz", then ".gz" is appended to the file name.

EXAMPLES

The following example shows how a SAIF file containing the user-annotated switching activity information on all design objects:

```
pt_shell> write_saif file
```

The following example shows how a SAIF file containing both user- annotated and propagated non-SDPD switching activity information can be generated:

```
pt_shell> write_saif file -propagated -exclude_sdpd
```

The following example shows how a SAIF file can be compressed.

```
pt_shell> write_saif file -compress gzip
```

The following example shows how a SAIF file on a synthesized design object can be generated by the **write_saif** command and read back using the **read_saif** command:

```
pt_shell> current_design top
pt_shell> current_instance U1
pt_shell> write_saif design.saif
pt_shell> reset_switching_activity
pt_shell> read_saif design.saif
```

SEE ALSO

```
get_switching_activity(2)
read_saif(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_rtl_to_gate_name(2)
set_switching_activity(2)
```

write_script

Writes design constraints as a script of commands for PrimeTime or Design Compiler.

SYNTAX

```
int write_script
   [-no_annotated_delay]
   [-no_annotated_check]
   [-format ptsh | dcsh | dctcl]
   [-output file_name]
   [-compress compression]
   [-include category_list]
   [-nosplit]
```

Data Types

file_name	string
compression	string
category_list	list

ARGUMENTS

-no_annotated_delay

Specifies that the **set_annotated_delay** commands are not to be written, which means that delay annotations are ignored.

-no_annotated_check

Specifies that the **set_annotated_check** commands are not to be written, which means that check annotations are ignored.

-format ptsh | dcsh | dctcl

Specifies the output format for the script. Valid values are **ptsh** (the default) for pt_shell, **dcsh** for dc_shell, and **dctcl** for dc_shell-t. Please note that the value **dcsh** has been made obsolete since the 16.12 release of Prime-Time and is no longer supported.

-output *file_name*

Specifies the name of a file to which output is to be written. The default is the standard output. If the **-compress** option is also specified, the extension ".gz" is added if it is not already present.

-compress *compression*

Specifies that the script is to be compressed. By default, the script is written as standard text. The only valid value for *compression* is **gzip**. If you specify this option, you must also specify the **-output** option.

-include *category_list*

Writes specified command categories only. The only valid value for *category_list* is **exceptions**.

-nosplit

Prevents line splitting. This is most useful for performing diffs on previous scripts or for postprocessing the script.

DESCRIPTION

This command writes design constraints as a script of commands for PrimeTime or Design Compiler (in dcsh or dctcl formats). The script is used to re-create design intent.

The **write_script** command writes the following information:

- Clock information: clock creation, generated clock creation, clock latency, clock uncertainty, interclock uncertainty.
- Timing information: disable timing, max time borrow. Disable timing set on library objects is written by default, but can be suppressed by setting the **write_script_include_library_constraints** shell variable to *false*.
- Point-to-point exceptions: false paths, min delay, max delay, multicycle paths, group path, input delay, output delay, annotated delay, and annotated checks.
- Net attributes: capacitance, resistance. These are no longer written by default. For more information, see the man page for the **write_script_output_lumped_net_annotation** shell variable.
- Port attributes: fanout, capacitance, and resistance.
- Design environment: wire load model, operating condition, drive, driving cell, and input transition.
- Design rules: minimum capacitance, maximum capacitance, minimum transition, maximum transition, minimum fanout, and maximum fanout.

EXAMPLES

The following example writes the script in ptsh format to the des.pt file:

```
pt_shell> write_script -output des.pt
```

SEE ALSO

```
characterize_context(2)
write_context(2)
write_script_output_lumped_net_annotation(3)
```

write_sdc

Writes out a script in Synopsys Design Constraints (SDC) format.

SYNTAX

```
int write_sdc
   file_name
   [-version sdc_version]
   [-compress compression]
   [-include categories_list]
   [-nosplit]
```

Data Types

```
file_name          string
sdc_version        string
compression        string
categories_list    list
```

ARGUMENTS

file_name

Specifies the name of the file to which the SDC script is to be written.

-version *sdc_version*

Specifies the version of SDC to write. Allowed values are **1.2**, **1.3**, **1.4**, **1.5**, **1.6**, **1.7**, **1.8**, **1.9**, **2.0**, and **latest** (the default).

-compress *compression*

Specify that the script should be compressed. The only valid value for *compression* is **gzip**.

-include *categories_list*

Write specified command categories only. The only valid value for *categories_list* is **exceptions**.

-nosplit

The **-nosplit** option prevents line-splitting. This is most useful for doing diff on previous scripts or for postprocessing the script.

DESCRIPTION

The **write_sdc** command writes out a script file in the latest Synopsys Design Constraints (SDC) format. This script contains commands that can be used with PrimeTime or with Design Compiler. SDC is also licensed by external vendors through the Tap-in program. SDC formatted script files are read into PrimeTime or Design Compiler using the **read_sdc** command.

Design Compiler supports **read_sdc** only in its Tcl mode, while the **write_sdc** command is available from either shell interface.

By default, the script is written as simple text. Optionally, the script can be compressed using the **-compress** option. In some cases, a file extension is appended to *file_name* if it is omitted. For example, with gzip formatted files, if *file_name* does not end with ".gz", then ".gz" is appended to the file name.

By default, the latest version of SDC is written. Some earlier versions of SDC can be written using the **-version** option.

SDC is a subset of the commands already supported by PrimeTime and Design Compiler. Of the commands supported in the latest SDC version, the following can be written by **write_sdc**. Those added for versions 1.3 and later are noted.

General Purpose Commands:

```
list
```

Object Access Functions:

```
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator
```

Basic Timing Assertions:

```
create_clock
create_generated_clock     (1.3)
set_clock_gating_check
set_clock_latency
set_clock_transition
set_clock_uncertainty
set_false_path
set_input_delay
set_max_delay
set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock
set_min_pulse_width        (2.0)
```

Secondary Assertions:

```
set_disable_timing
set_max_time_borrow
set_data_check             (1.4)
set_timing_derate          (1.5)
```

Environment Assertions:

```
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_load
set_max_area
set_max_capacitance
set_max_fanout
set_max_transition
set_min_capacitance
set_min_fanout
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
```

Similar to the **write_script** command, the **write_sdc** command writes out commands relative to the top of the design, regardless of the current instance. SDC files written by the **write_sdc** command must be read in from the top of the design.

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note*, which is available at http://solvnet.synopsys.com.

The usage of some of these commands is restricted when reading SDC. In some cases, some options are not allowed. The **write_sdc** command supports the restricted usage by restricting what is written. The following restriction apply for SDC Version 1.3 or later:

Note: For the **set_port_fanout_number** command, the **-min** and **-max** options are not supported.

When hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous - it is unclear which hierarchy separator characters are part of the name and which are real separators. Beginning with SDC version 1.2, hierarchical names can be made unambiguous using the **set_hierarchy_separator** SDC command or the **-hsc** option available from the **get_cells**, **get_lib_cells**, **get_lib_pins**, **get_nets**, and **get_pins** SDC object access commands.

By default, PrimeTime and Design Compiler writes an SDC file using these features to create unambiguous names. It is recommended that you write SDC files that contain unambiguous names. However, if you are using a third-party application that does not fully support SDC 1.2 or later (that is, it does not support the unambiguous hierarchical names features of SDC), you can suppress these features by setting the **sdc_write_unambiguous_names** variable to **true**.

The following features are added for SDC Version 1.5 or later:

- **-clock** option of the **set_clock_latency** command
- **set_timing_derate** command with all options
- **-clock_path -data_path -rise -fall** options of the **set_max_transition** command
- **-rise_from**, **-rise_to**, **-fall_from**, **-fall_to** options, of the **set_clock_uncertainty** command
- **-object_list** option of the **set_operating_conditions** command
- Synonyms for all of the **get_object** commands
- **get_objects -nocase** support independently with the **-regexp** option
- **get_objects -regexp** support

- **get_objects -of_objects** support for **get_cells/get_nets/get_pins**

The following features are added for SDC Version 2.0 or later:

- **-reference_pin** option of the **set_input_delay** and **set_output_delay** commands
- The **set_min_pulse_width** command

EXAMPLES

The following command writes the SDC script to the file top.sdc.

```
prompt> write_sdc top.sdc
```

SEE ALSO

```
read_sdc(2)
write_script(2)
sdc_write_unambiguous_names(3)
```

write_sdf

Writes a Standard Delay Format (SDF) back-annotation file.

SYNTAX

```
status write_sdf
   [-version sdf_version]
   [-no_cell_delays]
   [-no_timing_checks]
   [-no_net_delays]
   [-input_port_nets]
   [-output_port_nets]
   [-significant_digits digits]
   [-enabled_arcs_only]
   [-no_internal_pins]
   [-instance inst_name]
   [-context Verilog | VHDL | none]
   [-map sdf_map_file_list]
   [-annotated]
   [-levels level]
   [-no_edge]
   [-compress compression]
   [-include include_list]
   [-exclude exclude_list]
   [-no_negative_values values_list]
   [-no_edge_merging arc_type_list]
   [-delay_calculation_only_mode]
   [-exclude_cells cell_list]
   file_name
```

Data Types

sdf_version	string
digits	int
inst_name	string
sdf_map_file_list	list
level	int
compression	string
include_list	list
exclude_list	list
values_list	list
arc_type_list	list
cell_list	list
file_name	string

ARGUMENTS

-version *sdf_version*

Selects the SDF version to use. Supported SDF versions are **2.1** (default) and **3.0**.

-no_cell_delays

Indicates that no cell delays are be written in the SDF file. By default, all cell pin-to-pin delays are written to the SDF file. Cell delays include the load delay of the cell. Following the SDF conventions, only cell input pin to cell output pin delays are written. In case one cell output is unbuffered, delays are usually represented in libraries by a delay from an output pin to another output pin. Because this is not allowed by the SDF convention, the SDF delay for an unbuffered output is specified from cell inputs.

-no_timing_checks

Indicates that no cell timing checks are to be written in the SDF file. By default, all cell timing checks (for example, setup, hold, recovery, and removal) are written to the SDF file.

-no_net_delays

Indicates that no net delays are to be written in the SDF file. By default, all net pin-to-pin delays are written to the SDF file.

-input_port_nets

Indicates that the SDF file is to include delays of nets connected to input ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available. If the **-instance** option is specified, all net delays across the instance boundary leading to a pin inside the instance are included instead. The pin must be found on any of levels 1 to *level* of hierarchy if the **-levels level** option is specified.

-output_port_nets

Indicates that the SDF file is to include delays of nets connected to output ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available. If the **-instance** option is specified, then all net delays across the instance boundary leading from a pin inside the instance are included instead. The pin must be found on any of levels 1 to *level* of hierarchy if the **-levels** option is specified.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be written in SDF delay triplets. Allowed values are 0-13; the default is 3.

-enabled_arcs_only

Indicates that the SDF file is to contain delays only of enabled timing arcs, and is not to include delays of currently-disabled timing arcs. By default, delays of all timing arcs in the design are written to the SDF file, whether they are disabled or enabled.

-no_internal_pins

Indicates that the SDF file is not to include delay timing arcs from or to internal pins. Timing arcs to or from internal pins are expanded into delays from and to primary input and output of the given cell.

-instance *inst_name*

Specifies that the SDF file is to be written only for the instance named *inst_name*. By default, all pin names are relative to the *inst_name*. However, if boundary net delays are included (using the **-input_port_nets** or **-output_port_nets** option) all pin names are relative to the top design. Note that in general, if the **-input_port_nets** or **-output_port_nets** option is specified, boundary nets are written leaf-to-leaf and do not start or end on hierarchical pins. If boundary nets are required to start or end on hierarchical pins, see the **write_physical_annotations** command.

-context Verilog | VHDL | none

Specifies the context for writing bus names in SDF. Valid values are **Verilog**, **VHDL**, or **none** (the default). In the Verilog context, when pin names are displayed, the last two square bracket characters ("[" and "]") are not escaped. In the VHDL context, the last two parenthesis characters ("(" and ")") in a pin name are not escaped. In the default context none, all bus-delimiting characters are escaped with a backslash character ("\"). In any context, bus-delimiting characters in instance names are always escaped. When used with the **-map** option, the **-context** option also affects the way names are printed in mapped SDF files. In the Verilog context, names are printed in %s[%d] format; in the VHDL context, names are printed in %s(%d) format.

Note: Names are affected only if they are mapped using the bus(name_to_be_changed) function in the mapping file.

-map *sdf_map_file_list*

Specifies a list of mapping files the SDF writer is to use when writing out the SDF file. A mapping file contains a user-specified format for printing SDF cell delays and constraints. When writing out SDF for a cell, the SDF writer takes the user-specified mapping, if present, to print out SDF for the cell. If no user-specified mapping is present for a cell, the SDF writer writes out SDF in the normal way.

-annotated

Indicates that the SDF is to include only timing arcs that have been annotated with the **read_sdf**, **set_annotated_delay**, or **set_annotated_check** command.

-levels *level*

Specifies the number of levels of hierarchy for which the SDF is written out. Level 1 means only the top design or *inst_name*. Value of N means all levels of hierarchy, 1 to N. By default, all levels of hierarchy are written out. Note that boundary net delays (using the **-input_port_nets** or **-output_port_nets** options) typically have some net arcs from or to pins outside the *inst_name* option. The location of such outside pins is not limited by the **-levels** option. That is, the **-levels** and **-instance** options let you choose which boundary arcs are included, but do not restrict where the arcs lead outside of the *inst_name* option.

-no_edge

Indicates that the generated SDF is not to include any edges (posedge or negedge) for both combinational and sequential IOPATHs.

-compress *compression*

Specifies a format to be used to compress the file. The only valid value for *compression* is **gzip**. By default, files are not compressed.

-include *include_list*

Specifies a list of constructs and specifiers to include in the SDF file; you can specify one or more of the following:

- **SETUPHOLD** - Indicates that all SETUP and HOLD constructs are to be replaced by SETUPHOLD constructs. If a pair of setup and hold arcs are found between the same pin edges, timing information for the/both arc/arcs is written in a single SETUPHOLD construct. If a single setup/hold arc is found then the arc will be written in a single SETUPHOLD construct with no timing information for the hold/setup portion. SETUPHOLD supports negative values.

- **RECREM** - Indicates that all RECOVERY and REMOVAL constructs are to be replaced by RECREM constructs. If a pair of recovery and removal arcs are found between the same pin edges, timing information for both arcs is written in a single RECREM construct. If a single recovery/removal arc is found then the arc will be written in a single RECREM construct with no timing information for the removal/recovery portion. RECREM supports negative values and can be written only for version 3.0.

- **edge_specific_preset_clear** - Indicates that the generated SDF is to include the appropriate edges (posedge or negedge) for preset and clear arcs. The default behavior is not to include edge specifiers for preset and clear arcs.

-exclude *exclude_list*

Specifies a list of timing values of construct types to be either excluded from the SDF file in order to reduce its size, or to be replaced by another construct, as in the case of condelse. Allowed values are one or more of the following:

- **constant_nets** - Indicates that nets are to be omitted from the SDF file if they propagate a constant.
- **constant_delay_arcs** - Indicates that delay arcs are to be omitted from the SDF file if they propagate a constant, either from case analysis or logical inputs.
- **default_cell_delay_arcs** - Indicates that all default cell delay arcs are to be omitted from the SDF file if conditional delay arcs are present. If there are no conditional delay arcs, the default cell delay arcs are written to the SDF file.
- **wlm_load_delay** - Indicates that net delays and cell delays calculated using WLM are to be omitted from the SDF.
- **checkpins** - When Library Compiler finds both combinational and sequential arcs between pins, a checkpin is created so that all arcs are expanded in the db so that a single arc pinA->pinB is replaced by the combination of a positive unate arc pinA->pinAcheckpin1 with zero delay and an arc pinAcheckpin1->pinB with the same sense and values as the original arc. When this option is set the SDF is written out as if all checkpins were never created.
- **no_condelse** - Indicates that PrimeTime will not use the condelse statement to write out default iopaths. By default PrimeTime will replace default iopaths with the condelse construct. Specifying this option will result the condelse statement being replaced by a default iopath. This option should be used for generating simulator compatible SDF.
- **clock_tree_path_models** - Indicates that arcs which model clock tree paths are to be omitted from the SDF file.

-no_negative_values *values_list*

Specifies a list of timing value types whose negative values are to be zeroed out when writing to the SDF file. Allowed values for timing values are timing checks, cell delays and net delays. This option should be used when the SDF file is intended for simulator use. Using this option leads to inaccurate delay estimation in PrimeTime, so the user should use caution with this option.

-no_edge_merging *arc_type_list*

Specifies a list of arc types which are not to be compressed in the SDF file through edge merging. When this option is not selected PrimeTime merges the posedge/negedge arcs only if they are identical to within a tolerance of 0.00001 times the absolute value of the smaller delay. Allowed values are one or more of the following.

- **timing_checks** - Indicates that timing checks are not to be compressed in the SDF file through edge merging.
- **cell_delays** - Indicates that cell delays are not to be compressed in the SDF file through edge merging.

-delay_calculation_only_mode

This option is intended for speeding-up flows that use PrimeTime for delay calculation. The **write_sdf** command first checks if the design is in the updated state. If it is, it simply writes the already computed delays and leaves the design in the updated state, regardless of the presence of the **-delay_calculation_only_mode** option. If

the design is not in the updated state, by default, the command triggers the **update_timing** command, which brings the design in the updated state. However, if the **-delay_calculation_only_mode** option is used, PrimeTime does not trigger the **update_timing** command; it triggers only the operations that are necessary for writing the SDF file; the design is left in the not-updated state.

-exclude_cells *cell_list*

Specifies a list of cells which are not to be included in the SDF file. The net delays of the excluded cell's boundary pins are not excluded.

file_name

Specifies the name of the SDF file to be written.

DESCRIPTION

This command writes leaf cell pin-to-pin timing information to a disk file. Timing information is written in SDF format using versions v2.1 or v3.0. The timing file contains data associated with the netlist from which it is created.

By default, the file is written as simple text; you can use the **-compress** option to compress the file. In some cases, if you omit the file extension, the **write_sdf** command appends it to *file_name*. For example, for gzip formatted files, the **write_sdf** command appends .gz to the file name if .gz was not specified.

Use the **write_sdf** command only when the instance names in the design agree with the naming conventions of the system to which the timing file is written.

Timing information can be written in multiples of ns, ps, and us. The time unit in the SDF file is that specified in the logic library, and is written in the timing file under "timescale". If there is no time unit in the library, the unit is assumed to be a multiple of nanoseconds.

For efficiency, PrimeTime merges rise/fall delay triplets in SDF, if they are identical to within a very small tolerance.

EXAMPLES

The following example writes timing information for the *MULT16* design to a disk file called mult16.sdf, using SDF version 2.1:

```
pt_shell> write_sdf -version 2.1 mult16.sdf
```

You can use the **-level** option to write out interblock net delays. Assume a design with three hierarchical cells, h, h/h1, and h/h2; and two leaf cells, h/h1/u1 and h/h1/u2. The following command writes out all arcs that begin or end inside h but outside h/h1 and h/h2:

```
pt_shell> write_sdf -levels 1 -instance h h.sdf
```

The following example writes out the same arcs as the previous example, plus arcs that cross the boundary of h. Note that either the from or the to pin must be outside h and the other inside h. Therefore, the command does not write a feedthrough net through h on this particular design. Similarly, a net arc representing net h/h1/u1/Y -> h/h1/out -> h/h1/in -> h/h1/u2/A is not written out, because both pins h/h1/u1/Y and h/h1/u2/A are inside h/h1; that is, both are outside the region defined by the **-level 1 -instance h** option.

```
pt_shell> write_sdf -levels 1 -instance h -input_port_nets \
          -output_port_nets h.sdf
```

SEE ALSO

```
read_sdf(2)
remove_annotated_check(2)
remove_annotated_delay(2)
report_annotated_check(2)
report_annotated_delay(2)
set_annotated_check(2)
set_annotated_delay(2)
```

write_spice_deck

Writes to a SPICE deck the paths or arcs generated by the **get_timing_paths** or **get_timing_arcs** command.

SYNTAX

```
status write_spice_deck
   [-align_aggressors]
   [-analysis_type type]
   [-ground_coupling_capacitors]
   [-header header_file_name]
   [-initial_delay delay]
   [-logic_one_name v1name]
   [-logic_one_voltage v1]
   [-logic_zero_name v0name]
   [-logic_zero_voltage v0]
   [-minimum_transition_time trans]
   [-output file_name]
   [-sub_circuit_file spice_sub_circuit_file]
   [-sweep_size number_of_points]
   [-sweep_step num]
   [-sample_size number_of_samples]
   [-time_precision precision]
   [-transient_size tran_size]
   [-transient_step tran_step]
   [-use_probe]
   [-user_measures user_measure_list]
   [-variation]
   [-mc_sample_size number_of_mc_samples]
   paths_arcs_list
```

Data Types

type	list
header_file_name	string
delay	float
v1name	string
v1	float
v0name	strong
v0	float
trans	float
file_name	string
spice_sub_circuit_file	string
number_of_points	unsigned
num	float
number_of_samples	integer
precision	unsigned
tran_size	float
tran_step	float
user_measure_list	list
number_of_mc_samples	integer
paths_arcs_list	list

ARGUMENTS

-align_aggressors

Applies only to a *net* timing arc. The relative switching time of active aggressors on the net arc compute by the crosstalk delay or noise is written out in the corresponding piecewise linear (PWL) statement. It is effective only if the net has a coupled RC network annotated. The victim switches after the initial_delay and the active aggressors switches relative to that. SPICE uses the calculation engine to get the worst case alignment, and uses a similar setup so that relative alignment is valid. For example ,the filtered aggressors are not considered as effective during calculation and their coupling capacitances are grounded.

-analysis_type *type*

Specifies one of the following analysis types for the generated SPICE deck:

- Crosstalk analysis types: **max_rise**, **max_fall**, **min_rise**, and **min_fall**
- Noise analysis types: **above_high**, **above_low**, **below_high**, and **below_low**

This option has no effect on the timing path. The default for a timing arc is **max_rise**.

-ground_coupling_capacitors

Specifies that the aggressors of the timing path or timing arc are not written to the generated SPICE deck. The associated coupling capacitors are grounded with a factor one.

-header *header_file_name*

Specifies the path to the user header file whose content is copied to the generated SPICE deck. You can use this file to identify the SPICE deck, to include the library files, or to copy text to a SPICE deck for any other purposes to facilitate the SPICE run.

-initial_delay *delay*

Specifies the initial delay, in library unit, added to all PWL statements. The default is the longest clock period, or 1.0 library unit for asynchronous designs. Note: Setting *delay* to zero makes generating a ramp difficult and is not recommended.

-logic_one_name *v1name*

Specifies the name of the default upper rail voltage source. The default is VDD.

-logic_one_voltage *v1*

Specifies the upper rail of the voltage swing of the gate input pins. This is used in the PWL and voltage sources generated by the command. The default is main library voltage.

-logic_zero_name *v0name*

Specifies name of the default lower rail voltage source. The default is VSS.

-logic_zero_voltage *v0*

Specifies the lower rail of voltage swing of the gate input pins. The default is 0 volts.

-minimum_transition_time *trans*

Specifies the minimum transition time in nanoseconds (ns), to be used in all generated PWL if the transition time computed by PrimeTime is smaller than *trans*. The default is 0.001 ns. Transition times less than 0.0001 ns are

not recommended.

-output ***name***

If the **-sample_size** option is not used, this option specifies the name of the SPICE deck file to be written for the first timing path. SPICE deck files related to subsequent timing paths are also based on this name. If the **-sample_size** option is used, this option specifies the name of the directory to be created for writing the sampled SPICE deck files.

-sub_circuit_file ***spice_sub_circuit_file***

Specifies the path to the file that contains all the SPICE .subckt definitions of logic gates in the timing paths. By default, a subcircuit call uses the pin order in the Synopsys .lib file. Use this option if the SPICE subcircuit has a different pin order from that of the .lib file.

-sweep_size ***number_of_points***

Specifies the number of sweep points to be generated for each active aggressor of the net arc. The number of simulations increases geometrically with the number of the active aggressor. Use this option with the **-align_aggressors** option.

-sweep_step ***num***

Specifies the maximum time interval between sweep points generated for each active aggressor of the net arc. The unit is in nanoseconds. The default is 0.1 ns. Use this option with the **-align_aggressors** and **-sweep_size** options.

-sample_size ***number_of_samples***

Specifies the number of SPICE deck files that have to be created while performing variation-aware timing analysis. This option takes the name of the directory through the **-output** option and creates multiple SPICE deck files that correspond to various samples of the variations defined.

-time_precision ***precision***

Specifies the number of precision digits for time in the PWL generated. The default is 6, and the range is from 1 to 20.

-transient_size ***tran_size***

Specifies the total transient time used in the SPICE .tran statement. The unit is in nanoseconds.

-transient_step ***tran_size***

Specifies the transient step size used in the SPICE .tran statement. The unit is in nanoseconds. The default is 0.001 ns.

-use_probe

Uses the .probe statement to output the node voltage instead of the .print statement.

-user_measures ***user_measure_list***

Uses the user-specified measures instead of the ones generated automatically by the SPICE deck. To remove all automatically generated .measure and .print statements from the SPICE deck, specify an empty *user_measure_list*.

-variation

Generate SPICE deck for Monte Carlo based simulation for path arrival. The number of Monte Carlo samples follows **-mc_sample_size** option. This option only works when *paths_arcs_list* is path object.

-mc_sample_size ***number_of_mc_samples***

Specifies the number of Monte Carlo samples in SPICE deck. By default, *number_of_mc_samples* is 5000. This option has impact only when **-variation** option is specified.

paths_arcs_list

Specifies the collection of timing paths or timing arcs that their circuits are written out.

DESCRIPTION

This command writes out a number of SPICE decks, one file for each timing path or timing arc from the collection generated by the **get_timing_paths** or **get_timing_arcs** command. For example, if you specify the **-output timing_path.spo** option for the first timing path, the second file name is timing_path_00001.spo, the third file is timing_path_00002.spo, and so on. The files contain the circuit generated and the side pin voltages. The **write_spice_deck** command also puts the corresponding stimulus, such as PWL and PULSE statements, in a separate stimulus files. The example of the stimulus file names are timing_path_stim.spo, timing_path_00001_stim.spo, …, etc. The stimulus file is included by the main circuit file by the SPICE .include statement.

If you specify a subcircuit file using the **-sub_circuit_file** option, all lines except .include and .subckt are ignored. The file on the .include line is further opened and parsed for other .include and .subckt lines. The pin order of each .subckt line is recorded and applied when a subcircuit call is generated. If a gate used is not in the subcircuit file, the .lib pin order is used. A warning is generated if a pin in the subcircuit definition is not found in the .lib file. This mismatched pin is written out to the SPICE deck in the same order as it appears in the subcircuit file. An error message is generated when a .lib gate pin is not found in the subcircuit definition.

Note that side input pins are sensitized locally.

The voltage used in PWL and PULSE statements depend on the pin that the voltage source is attached to unless the **library_thresholds_use_main_lib** variable is set to **true**. You cannot specify multiple rails.

The timing PWL voltage sources are defined by the **set_library_driver_waveform** command. The default is Synopsys pre-driver. The **set_library_driver_waveform** command defines the shape of the waveform the timing library is characterized with.

The **-align_aggressors** option applies only to active aggressors of the net timing arcs. The non-active or filtered aggressors are set to be quiet and their coupling capacitors grounded. To find the active aggressors of a victim, use **report_delay_calculation -crosstalk** or **report_noise_calculation**. While the **-aligned_aggressors** option is set, the **-sweep_size** option with more than zero sweeping points writes out the PWL in sweep form. A third file with the sweep data is generate. An example of the name of this file is timing_arc_sweep.spo. It is also included by the main circuit file. It must be noted that under most circumstances, the **-align_aggressors** option writes out the alignment to produce the worst case stage delay. However, under some corner cases, for instance when the crosstalk bump is very large, the **-sweep** option should be used in conjunction to obtain the worst case alignment.

The sweep points center around the aligned aggressor switching time generated without the **-sweep_size** option. If an even number is given, one more sweep point on the smaller switch time side is generated. The time difference between the sweep points are controlled by the **-sweep_step** and **-sweep_size** options and the aggressor sweep range is computed by the crosstalk delay engine. The lesser of the (sweep_range)/(number_of_sweep_points-1) for each aggressor and the **-sweep_step** option is used.

Be very careful with the number of points specified by the **-sweep_size** option. The number of SPICE simulation grows geometrically larger as the number of the active aggressors increases. For example, three sweep points for five active aggressors results in 3^5 (243) simulations.

Use the **-user_measures** option to create specific measure statements instead of automatic generated measures. The *user_measure_list* argument is a list of user-defined measures. Each user-defined measure can be one of the following types,

```
{delay from_pin/port_name to_pin/port_name [meas_name]}
{slew pin_or_port_name [meas_name]}
{noise_peak pin_or_port_name [meas_name]}
{user_string "user_defined_string"}
```

The **write_spice_deck** command uses these user-defined measure instructions and creates SPICE .measures with proper trip points and switching directions. The SPICE measures are in the same order as the measure generated when using the *user_measure_list* option. When this option is provided, the user-defined measure overrides the automatic .measure and .print statements.

You could give zero or more measures in the list. When there is no user-defined measure in the list, for example, when **-user_measures { }**, SPICE deck does not have any kind of .measure and .print statements. You can also insert user-specific measures in the user_string. The user-defined string is inserted as it is. The **write_spice_deck** command does not verify the validity of the user string. For the **delay**, **slew**, and **noise_peak** types, you can give a name to the measurement as *meas_name*. If it is missing, the **write_spice_deck** command automatically generates a name of the measurement. If the user-defined measure does not match with any of the previous format, the command fails.

If you specify the **-sample_size** option while performing variation aware timing analysis, a directory is created and multiple SPICE deck files are written with sampled values inside it.

Note that this command is not compatible with the **timing_reduce_parallel_cell_arcs** variable. This variable should be set to **false** to ensure proper parallel cell arc sensitization.

To use the **write_spice_deck** command, you must have a PrimeTime SI license.

EXAMPLES

The following example writes a SPICE deck for timing paths obtained by **get_timing_paths**, using the my_subckt.sp file as the source of all SPICE .subckt definitions of all gates in the timing paths. The other two examples are for timing arcs.

```
pt_shell> write_spice_deck -output path.spo \
            -sub_circuit_file my_subckt.sp [get_timing_paths]

pt_shell> write_spice_deck -output path.spo \
            -analysis_type above_low \
            -sub_circuit_file my_subckt.sp [get_timing_arcs -from U1/A]

pt_shell> write_spice_deck -output ./arc.spi \
            -analysis_type max_rise \
            -align_aggressors -sweep_size 3 -sweep_step 0.01 \
              [get_timing_arc -from I1_1/Z -to I1_2/A ]
```

This example shows how to generate measure statement to measure the slew at the path output and delay from input to output.

```
pt_shell> write_spice_deck [get_timing_path -from [get_port i2] \
            -to [get_port o2]] -user_measures { {slew o2 my_out_slew } \
            {delay i2 o2 path_delay} {user_string ".print o2"} }

        .measure tran my_out_slew
        + trig v(o2) val = 2.16 td = 7.5e-09 fall = 1
        + targ v(o2) val = 0.54 td = 7.5e-09 fall = 1
```

```
        .measure tran path_delay
        + trig v(i2) val = 1.35 td = 7.5e-09 fall = 1
        + targ v(o2) val = 1.35 td = 7.5e-09 fall = 1
        .print o2
1
```

SEE ALSO

```
get_timing_arcs(2)
get_timing_paths(2)
report_delay_calculation(2)
report_noise_calculation(2)
set_library_driver_waveform(2)
timing_reduce_parallel_cell_arcs(3)
```

write_taps

Writes information on the currently inserted tap points into a text file that can be used to re-insert the tap points in another session.

SYNTAX

```
status write_taps
   file_name
   [taps]
   [-replace]
```

Data Types

```
string file_name
list taps
```

ARGUMENTS

file_name

Specifies the name of the file to which the tap point insertion script is to be written.

taps

Specifies the tap points to be written. The tap points can be specified as name patterns or collections of taps.

-replace

Overwrites the existing *file_name* tap file. If the *file_name* tap file already exists and the *-replace* option is not specified, the command returns a failure.

DESCRIPTION

The command writes a file containing the location and associated supply net for each inserted tap, which can be read into a later session with the **create_taps -import file** command.

The tap points written to the file can be limited to specific types by using the filter option with the **get_taps** command. For example, you can capture the set of tap points that were automatically inserted into the session and were of *auto_pg* type if the **rail_enable_auto_pg_taps** variable is set to true.

The following is the format of the output file:

supply_net_name Milkyway_layer_number X_coordinate Y_coordinate [R(R_value)] [L(L_value)] [C(C_value)]

Lumped RLC values will be written to the tap file if they are specified for the taps. The units are Ohm for resistance, Henry for inductance, and Farad for capacitance. In the following example, no lumped RLC value is written to the tap file:

```
VDDY 39 746.800 469.400
VDDY 39 0.200 469.400
VDDY 39 0.200 548.600
VDDY 39 0.200 627.800
VSS 39 746.800 51.800
VSS 39 0.200 51.800
VSS 39 746.800 131.000
VSS 39 746.800 210.200
VSS 39 746.800 368.600
```

In the following example, the lumped RLC value is written to the tap file:

```
VDDY 39 746.800 469.400 R(0.001) L(1.0e-9) C(2.0e-12)
VDDY 39 0.200 469.400
VDDY 39 0.200 548.600
VDDY 39 0.200 627.800
VSS 39 746.800 51.800
VSS 39 0.200 51.800
VSS 39 746.800 131.000
VSS 39 746.800 210.200
VSS 39 746.800 368.600
```

EXAMPLES

The following example writes a file containing the taps that were automatically generated by the **create_taps** command and those inserted via the lib_cell reference. A previously generated tap file is to be replaced.

```
pr_shell> write_taps [get_taps -filter type {auto_pg libcell_pg}] -replace ./MyTapFile
```

SEE ALSO

create_taps(2)
remove_taps(2)
report_taps(2)
get_taps(2)