

PrimeTime® Suite

Tool Commands

Version J-2014.12, December 2014

SYNOPSYS®

Copyright Notice and Proprietary Information

© 2014 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

Synopsis and certain Synopsis product names are trademarks of Synopsis, 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. Synopsis does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

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

Copyright Notice for the Command-Line Editing Feature

© 1992, 1993 The Regents of the University of California. All rights reserved. This code is derived from software contributed to Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright Notice for the Line-Editing Library

© 1992 Simmule Turner and Rich Salz. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The authors are not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software.
Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

Copyright Notice for the jemalloc Memory Allocator

© 2002-2013 Jason Evans <jasone@canonware.com>. All rights reserved.

© 2007-2012 Mozilla Foundation. All rights reserved.

© 2009-2013 Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice(s), this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice(s), this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER(S) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND

ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright Notice for the CDPL Common Module

© 2006-2014, Salvatore Sanfilippo. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Redis nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

Collections_and_Querying	1
add_to_collection	6
alias	9
all_clocks	11
all_connected	12
all_fanin	14
all_fanout	17
all_inputs.....	20
all_instances.....	22
all_outputs	24
all_registers.....	26
append_to_collection	30
apropos.....	32
associate_supply_set	34
cell_of.....	35
change_selection	36
characterize_context.....	39
check_block_scope.....	41
check_constraints.....	44
check_level_shifters	46
check_noise	47
check_power.....	50
check_timing.....	52
compare_collections.....	60
compare_interface_timing.....	62

complete_net_parasitics	68
connect_net	70
connect_power_domain	72
connect_power_net_info	74
connect_supply_net	76
copy_collection	78
cputime	80
create_cell	81
create_clock	85
create_command_group	88
create_generated_clock	89
create_histogram	94
create_ilm	99
create_merged_modes	106
create_net	110
create_operating_conditions	113
create_placement_blockage	115
create_power_domain	117
create_power_group	121
create_power_net_info	123
create_power_rail_mapping	125
create_power_switch	128
create_qtm_constraint_arc	131
create_qtm_delay_arc	134
create_qtm_drive_type	137
create_qtm_generated_clock	140
create_qtm_load_type	142
create_qtm_model	144
create_qtm_path_type	146
create_qtm_port	149
create_scenario	151
create_si_context	154
create_supply_net	157
create_supply_port	159
create_supply_set	161
create_voltage_area	165

current_design	168
current_instance	170
current_power_rail	173
current_scenario	175
current_session	178
date	180
define_design_mode_group	181
define_name_maps	183
define_proc_attributes	186
define_qtm_attribute	190
define_scaling_lib_group	192
define_user_attribute	194
derive_clocks	197
disconnect_net	199
drive_of	201
echo	203
enable_primestime_icc_consistency_settings	204
error_info	205
estimate_clock_network_power	206
estimate_eco	208
exit	215
extract_model	216
filter	221
filter_collection	222
find	225
find_objects	227
fix_eco_drc	229
fix_eco_leakage	238
fix_eco_power	244
fix_eco_timing	251
foreach_in_collection	267
get_alternative_lib_cells	269
get_app_var	272
get_attribute	275
get_cells	277
get_clock_network_objects	280

get_clocks.....	282
get_command_option_values.....	284
get_current_power_domain	286
get_current_power_net.....	287
get_defined_attributes	289
get_defined_commands.....	292
get_designs	295
get_distributed_variables	298
get_generated_clocks.....	302
get_ilm_objects.....	304
get_latch_loop_groups	305
get_lib_cells	307
get_lib_pins.....	310
get_lib_timing_arcs.....	313
get_libs	316
get_license	319
get_message_ids	320
get_message_info	321
get_nets	323
get_noiseViolation_sources.....	327
get_object_name	330
get_path_groups.....	331
get_pins	333
get_ports.....	337
get_power_domains	340
get_power_group_objects.....	342
get_power_per_pgpin	343
get_power_switches	344
get_qtm_ports	346
get_random_numbers.....	348
get_selection.....	349
get_si_bottleneck_nets	352
get_supply_nets	354
get_supply_ports	356
get_supply_sets	358
get_switching_activity.....	360

get_timing_arcs.....	368
get_timing_paths	372
get_unix_variable	382
getenv.....	383
group_path	385
gui_create_category_rule	389
gui_get_category	392
gui_get_cell_block_marks.....	394
gui_list_category_rules	396
gui_list_cell_block_marks.....	399
gui_remove_category_rules	400
gui_remove_cell_block_marks	402
gui_set_cell_block_marks.....	404
gui_start	407
gui_stop	408
gui_write_window_image	409
help.....	411
help_attributes	413
history.....	414
identify_interface_logic	418
index_collection	423
infer_switching_activity.....	425
insert_buffer	427
is_false	436
is_true.....	438
license_users	440
link	441
link_design	442
list_attributes.....	447
list_designs.....	450
list_key_bindings.....	452
list_libs	453
list_licenses.....	455
lminus	456
load_of	458
load_upf	459

ls	461
man	462
map_design_mode	463
mem	465
merge_models	466
merge_saif	469
parallel_execute	472
parallel_foreach_in_collection	475
parse_proc_arguments	480
post_eval	482
print_message_info	484
print_suppressed_messages	486
printenv	487
printvar	488
proc_args	490
proc_body	491
query_cell_instances	492
query_cell_mapped	493
query_net_ports	494
query_objects	495
query_port_net	498
quit	499
read_aocvm	500
read_db	502
read_ddc	504
read_file	506
read_lib	508
read_milkyway	510
read_parasitics	513
read_saif	520
read_sdc	523
read_sdf	529
read_vcd	534
read_verilog	538
read_vhdl	542
redirect	544

remote_execute	548
remove_annotated_check.....	550
remove_annotated_clock_network_power.....	554
remove_annotated_delay.....	555
remove_annotated_parasitics.....	557
remove_annotated_power	559
remove_annotated_transition.....	561
remove_aocvm.....	563
remove_buffer.....	565
remove_capacitance.....	567
remove_case_analysis	569
remove_cell.....	571
remove_clock	573
remove_clock_gating_check.....	575
remove_clock_groups.....	577
remove_clock_latency	579
remove_clock_sense	581
remove_clock_transition.....	582
remove_clock_uncertainty	583
remove_connection_class	586
remove_context	587
remove_coupling_separation	589
remove_current_session.....	591
remove_data_check	592
remove_design.....	594
remove_design_mode	596
remove_disable_clock_gating_check.....	598
remove_disable_timing.....	599
remove_drive_resistance	601
remove_driving_cell	603
remove_fanout_load.....	605
remove_from_collection	606
remove_generated_clock.....	608
remove_host_options	609
remove_ideal_latency.....	615
remove_ideal_network	617

remove_ideal_transition	619
remove_input_delay	621
remove_input_noise	623
remove_lib	624
remove_license	626
remove_max_area	627
remove_max_capacitance	628
remove_max_fanout	629
remove_max_time_borrow	630
remove_max_transition	631
remove_min_capacitance	633
remove_min_pulse_width	635
remove_multi_scenario_design	637
remove_net	638
remove_noise_immunity_curve	639
remove_noise_lib_pin	641
remove_noise_margin	642
remove_operating_conditions	643
remove_output_delay	644
remove_parasitic_corner	646
remove_path_group	647
remove_port_fanout_number	648
remove_power_groups	649
remove_propagated_clock	650
remove_pulse_clock_max_transition	651
remove_pulse_clock_max_width	653
remove_pulse_clock_min_transition	654
remove_pulse_clock_min_width	655
remove_qtm_attribute	656
remove_rail_voltage	658
remove_resistance	659
remove_scenario	660
remove_sense	661
remove_setup_hold_pessimism_reduction	663
remove_si_aggressor_exclusion	664
remove_si_delay_analysis	666

remove_si_delay_disable_statistical	669
remove_si_noise_analysis	670
remove_si_noise_disable_statistical	673
remove_steady_state_resistance	674
remove_user_attribute	675
remove_waveform_integrity_analysis	677
remove_wire_load_min_block_size	678
remove_wire_load_model	679
remove_wire_load_selection_group	681
rename	682
rename_cell	684
rename_design	686
rename_net	688
report_activity_file_check	690
report_activity_waveforms	695
report_alternative_lib_cells	697
report_analysis_coverage	702
report_annotated_check	708
report_annotated_delay	711
report_annotated_parasitics	714
report_annotated_power	718
report_aocvm	720
report_app_var	724
report_attribute	725
report_bottleneck	728
report_bus	732
report_case_analysis	733
report_cell	736
report_cell_usage	740
report_clock	743
report_clock_gate_savings	748
report_clock_gating_check	754
report_clock_timing	756
report_constraint	769
report_context	781
report_crpr	783

report_delay_calculation	789
report_design	801
report_design_mismatch	803
report_disable_timing	805
report_driver_model	808
report_eco_library_cells	811
report_eco_options	814
report_etm_arc	816
report_exceptions	820
report_global_slack	825
report_global_timing	828
report_hierarchy	836
report_host_usage	838
report_ideal_network	847
report_latch_loop_groups	850
report_lib	852
report_lib_groups	858
report_min_period	860
report_min_pulse_width	864
report_mode	868
report_multi_input_switching_coefficient	872
report_multi_scenario_design	873
report_name_mapping	876
report_net	878
report_noise	881
report_noise_calculation	885
report_noise_parameters	889
report_noiseViolation_sources	890
report_ocvm	893
report_path_group	895
report_port	897
report_power	900
report_power_analysis_options	916
report_power_calculation	917
report_power_derate	923
report_power_domain	926

report_power_groups	929
report_power_net_info	931
report_power_network	933
report_power_pin_info	934
report_power_rail_mapping	937
report_power_switch.	939
report_pulse_clock_max_transition	940
report_pulse_clock_max_width.	943
report_pulse_clock_min_transition	945
report_pulse_clock_min_width	947
report_qor	949
report_qtm_model.	953
report_reference	956
report_scale_parasitics.	958
report_scope_data	959
report_sense.	962
report_si_aggressor_exclusion.	964
report_si_bottleneck	967
report_si_delay_analysis	972
report_si_double_switching	977
report_si_noise_analysis	979
report_supply_net.	984
report_supply_set	986
report_switching_activity.	988
report_threshold_voltage_group.	1000
report_timing.	1003
report_timing_derate.	1024
report_transitive_fanin	1028
report_transitive_fanout	1030
report_units.	1033
report_vcd_hierarchy	1035
report_waveform_integrity_analysis	1037
report_wire_load.	1039
reset_aocvm_table_group	1041
reset_design.	1042
reset_eco_options	1043

reset_mode	1044
reset_multi_input_switching_coefficient	1047
reset_noise_parameters	1048
reset_ocvm_table_group	1049
reset_path	1051
reset_power_derate	1055
reset_rtl_to_gate_name	1057
reset_scale_parasitics	1058
reset_switching_activity	1059
reset_timing_derate	1063
restore_session	1066
save_qtm_model	1068
save_session	1070
scale_parasitics	1074
set_active_clocks	1076
set_annotated_check	1078
set_annotated_clock_network_power	1082
set_annotated_delay	1085
set_annotated_power	1089
set_annotated_transition	1091
set_aocvm_coefficient	1093
set_aocvm_table_group	1095
set_app_var	1097
set_case_analysis	1099
set_clock_gating_check	1101
set_clock_groups	1104
set_clock_latency	1108
set_clock_sense	1112
set_clock_transition	1113
set_clock_uncertainty	1115
set_connection_class	1119
set_context_margin	1121
set_coupling_separation	1123
set_cross_voltage_domain_analysis_guardband	1125
set_current_command_mode	1127
set_current_power_domain	1129

set_current_power_net	1132
set_data_check	1136
set_design_attributes	1139
set_design_top	1141
set_disable_clock_gating_check	1142
set_disable_timing	1144
set_distributed_parameters	1148
set_distributed_variables	1150
set_domain_supply_net	1152
set_dont_touch	1154
set_dont_touch_network	1157
set_dont_use	1159
set_drive	1161
set_drive_resistance	1164
set_driving_cell	1166
set_eco_options	1170
set_equal	1174
set_false_path	1176
set_fanout_load	1180
set_host_options	1181
set_ideal_latency	1186
set_ideal_network	1188
set_ideal_transition	1191
set_input_delay	1193
set_input_noise	1198
set_input_transition	1200
set_isolation	1202
set_isolation_control	1206
set_latch_loop_breaker	1208
set_level_shifter_strategy	1210
set_level_shifter_threshold	1211
set_lib_rail_connection	1213
set_library_driver_waveform	1214
set_load	1216
set_max_area	1220
set_max_capacitance	1221

set_max_delay	1224
set_max_fanout	1229
set_max_time_borrow.	1231
set_max_transition	1233
set_message_info.	1236
set_min_capacitance	1238
set_min_delay.	1240
set_min_library	1245
set_min_pulse_width	1247
set_mode	1250
set_multi_input_switching_coefficient.	1253
set_multi_scenario_license_limit	1255
set_multicycle_path	1257
set_noise_derate	1264
set_noise_immunity_curve.	1266
set_noise_lib_pin	1269
set_noise_margin	1270
set_noise_parameters	1272
set_ocvm_table_group	1274
set_operating_conditions	1276
set_opposite	1281
set_output_delay	1283
set_parasitic_corner	1288
set_port_attributes	1290
set_port_fanout_number.	1293
set_power_analysis_options.	1295
set_power_clock_scaling	1300
set_power_derate.	1302
set_program_options	1306
set_propagated_clock.	1309
set_pulse_clock_max_transition.	1311
set_pulse_clock_max_width	1313
set_pulse_clock_min_transition	1315
set_pulse_clock_min_width	1317
set_qtm_attribute	1319
set_qtm_global_parameter.	1321

set_qtm_port_drive	1323
set_qtm_port_load	1325
set_qtm_technology	1327
set_query_rules	1330
set_rail_voltage	1335
set_related_supply_net.	1338
set_resistance.	1340
set_retention.	1342
set_retention_control	1345
set_rtl_to_gate_name.	1347
set_scope	1349
set_sense	1351
set_setup_hold_pessimism_reduction	1353
set_si_aggressor_exclusion	1355
set_si_delay_analysis.	1357
set_si_delay_disable_statistical	1360
set_si_noise_analysis.	1361
set_si_noise_disable_statistical	1364
set_steady_state_resistance	1365
set_switching_activity	1367
set_temperature	1373
set_timing_derate	1375
set_units	1382
set_unix_variable	1384
set_user_attribute.	1385
set_voltage	1387
set_waveform_integrity_analysis	1390
set_wire_load_min_block_size	1392
set_wire_load_mode.	1393
set_wire_load_model	1394
set_wire_load_selection_group	1396
setenv.	1398
sh	1400
sh_list_key_bindings.	1402
sim_analyze_clock_network.	1403
sim_enable_si_correlation	1405

sim_setup_library	1406
sim_setup_simulator.....	1408
sim_setup_spice_deck	1410
sim_validate_path.....	1411
sim_validate_setup.....	1413
sim_validate_stage.....	1415
size_cell	1417
sizeof_collection	1421
sort_collection.....	1423
source.....	1425
start_gui	1427
start_hosts	1428
start_profile.....	1434
stop_gui	1436
stop_hosts	1437
stop_profile.....	1443
suppress_message.....	1444
swap_cell	1445
transform_exceptions	1448
unalias	1453
unset_rtl_to_gate_name.....	1454
unsetenv	1455
unsuppress_message.....	1457
update_noise	1458
update_power.....	1460
update_scope_data	1463
update_timing	1465
upf_version	1466
which.....	1468
write_activity_waveforms	1469
write_app_var.....	1473
write_arrival_annotations	1474
write_binary_aocvm	1476
write_changes.....	1478
write_context.....	1483
write_ilm_netlist	1486

write_ilm_parasitics	1488
write_ilm_script	1490
write_ilm_sdf	1492
write_interface_timing	1494
write_parasitics	1497
write_physical_annotations	1499
write_saif	1502
write_script	1504
write_sdc	1506
write_sdf	1510
write_sdf_constraints	1516
write_spice_deck	1520

Collections_and_Querying

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 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  

```

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)
```

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

<i>collection1</i>	collection
<i>object_spec</i>	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.

add_to_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

<i>name</i>	string
<i>def</i>	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":

```
all_connected
```

```
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]
[-step_into_hierarchy]
[-levels level_count]
[-pin_levels pin_count]
[-trace_arcs timing | enabled | all]
```

Data Types

<i>from_list</i>	list
<i>through_list</i>	list
<i>to_list</i>	list
<i>level_count</i>	int
<i>pin_count</i>	int

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.

-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 *level_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.
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 **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 that 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]
  [-step_into_hierarchy]
  [-levels level_count]
  [-pin_levels pin_count]
  [-trace_arcs timing | enabled | all]
```

Data Types

<i>from_list</i>	list
<i>through_list</i>	list
<i>to_list</i>	list
<i>level_count</i>	int
<i>pin_count</i>	int

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

Indicates that all clock source pins and ports in the design are to be used as the list of **from** pins. Clock sources are specified using **create_clock**. If there are no clocks, or if the clocks have no sources, the result is the empty

collection. This option is exclusive with the **-from** option. Additionally, the **-clock_tree** option constrains the search to objects in the clock network only.

-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.

-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.
 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 **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 the **-clock_tree** option is used, the search is constrained to objects in the clock network only.

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 a register, reg3.

```
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 *from_list* is an output pin of a hierarchical cell, H3/z1, which is connected to an input pin of another hierarchical cell, H4/a. H4 contains a leaf cell U1 with input A and output Z. The first command is 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

<i>clock_name</i>	list
<i>rise_clock_name</i>	list
<i>fall_clock_name</i>	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]
```

append_to_collection

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)
```

apropos

32

```
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

<i>supply_set_handle</i>	string
<i>supply_set_name</i>	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)
```

associate_supply_set

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_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

<i>slct_bus</i>	string
<i>object_type</i>	string
<i>clock_tree_list</i>	list
<i>collection</i>	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
[-timing]
[-environment]
[-design_rules]
[-constant_inputs]
[-no_boundary_annotations]
cell_list
```

Data Types

cell_list list

ARGUMENTS

-timing
Characterizes timing information; for example, clocks, input and output delays, and timing exceptions.

-environment
Characterizes environment-related information; for example, 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
Characterizes design rules; for example, max_capacitance, max_transition, and max_fanout.

-constant_inputs
Characterizes logic constants propagated to input pins of the instance being characterized by the case analysis capability of PrimeTime.

-no_boundary_annotations
Disables characterization of annotated capacitance on boundary nets as annotated capacitance in the characterized instance. Instead, the port wire capacitance is adjusted to account for any difference between the estimated and annotated values. By default, PrimeTime characterizes annotated capacitance on boundary nets as annotated capacitance in the characterized instance.

cell_list
Specifies a list of instances to characterize.

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, wire load models, input drives, and capacitive loads.

The **characterize_context** command does not capture delays and parasitics annotated on the internal nets of the instance being characterized. To capture this information, use the **write_physical_annotations** command without the **-boundary_nets** option.

All categories are characterized if no option is specified. To characterize one or more categories of interest, specify the respective option or list of options.

EXAMPLES

The following command derives only the timing-related context information for instance I1 and I2.

```
pt_shell> characterize_context -timing {I1 I2}
```

The following command derives the design rule and the logic constant information from the context of cell I2.

```
pt_shell> characterize -constant_inputs -design_rules I2
```

SEE ALSO

```
remove_context(2)
report_context(2)
write_context(2)
write_physical_annotations(2)
```

check_block_scope

Checks the scope of hierarchical blocks that were replaced with timing models during the top-level analysis.

SYNTAX

```
int check_block_scope
-instances instance_list
[-scope_scenario scenario_name]
[-check_types check_types]
[-absolute_clock_arrival]
[-significant_digits digits]
[-nosplit]
[-verbose]
file_name
```

Data Types

instance_list	list
scenario_name	string
check_types	string
digits	int
file_name	string

ARGUMENTS

- instances *instance_list*
Specifies the list of instances for which scope needs to be verified. The scope information for the blocks corresponding to these instances should be contained in the file specified by the *file_name* option.
- scope_scenario *scenario_name*
Specifies the scenario name for which the scope data needs to be used to verify the top-level instantiation of the block as timing models. The scope information in the data file should contain data corresponding to the given scenario. If you do not specify, a fixed default name is used internally.
- check_types *check_types*
Specifies the types of scope checks to be performed. By default, when this option is not specified, the types of checks defined by the *hier_scope_check_defaults* environment variable are performed. Use this option to overrides the globally defined default scope check types. The available check types are *clock_arrival*, *clock_transition*, *data_input_arrival*, *data_input_transition*, *clock_uncertainty*, and *clock_skew_with_uncertainty*. When transparent latches are used on the block interface paths, it is important to verify the block scope using the *data_input_arrival* check, in addition to the default checks.
- absolute_clock_arrival
Indicates that the clock arrival times reach the block boundary pins are to be checked as absolute rather than relative latency values against those captured as scope information during original block-level analysis. By

default, when this option is not used, the arrival windows for clock signals are checked as relative arrival requirements. For more information, see the DESCRIPTION section.

-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.

-nosplit

Specifies that the printed report should not split the lines even when they become long and unfit for the line column number settings. By default, the report lines are split, when necessary.

file_name

Specifies the file name containing information to load the scope data.

-verbose

Displays a verbose report of the scope check results.

DESCRIPTION

This command should be used while performing top-level analysis in a bottom-up hierarchical design flow. At the block-level, timing analysis has been performed and a timing model is created through the **create_ilm** or **extract_model** command. In addition, scope data can be generated for the models by specifying appropriate options to these two commands. The **check_block_scope** command checks the scope of block instances to make sure that the timing model accurately reflects the full block.

The checks that are performed are dictated by the **hier_scope_check_defaults** variable.

For a given block, when multiple clocks interact with each other and, therefore, should be maintained synchronous to each other, it indicates that the relative skews among these clocks should be maintained to safely move the analysis from block-level to top-level. It is not necessary to require for each and every clock that the absolute arrival times at the block boundary for all instances of the block remain unchanged. It is sufficient for all these clocks to incur a similar amount of insertion delay when propagate through their top-level clock trees before reaching the block boundary as long as their inter-clock skew relations are maintained. This is also applicable to blocks with only one clock.

When the **-absolute_clock_arrival** option is not specified, PrimeTime automatically derives a best common shift, separately for each synchronous clocking domain and each instance of the block need to be checked, based on the original arrival windows captured into the scope file and the actual arrival windows obtained at top-level. This global shift is then uniformly applied across all the clocks in that domain such that the number of scope violations reported is minimized. Note that for the same block, different global shift might be derived for different instances. The

goal of deriving and applying the global shift is to minimize the scope violations reported and therefore need to be fixed potentially. It may or may not correlate directly to the actual common insertion delay for all the related clocks in that domain of an instance of the block.

This flow can be used in both PrimeTime and PrimeTime SI.

EXAMPLES

This example uses the scope information saved in top/ilm.scope file to verify that the timing models for the I1 and I2 instances are within the validated scope of the block.

```
pt_shell> check_block_scope -instances {I1 I2} top/ilm.scope
```

SEE ALSO

```
create_ilm(2)
extract_model(2)
create_scenario(2)
hier_scope_check_defaults(3)
```

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    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    0    Input/inout port 'A' has no input delay specified
  2 of 4                  0    0    Input/inout port 'B' has no input delay specified
  3 of 4                  0    0    Input/inout port 'C' has no input delay specified
  4 of 4                  0    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    0    Output/inout port 'QA' has no clock-
related output delay specified
```

check_constraints

```
    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}`.

SYNTAX

```
int 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_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

<i>digits</i>	int
<i>check_list</i>	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 using check_list. See the man
    page of power_check_defaults for its default value.

-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.
```

check_list
Gives the list of checks to be performed. Each element in this list is one
of the following strings: no_arrival_window, no_base_clock,
out_of_table_range, missing_table, missing_function.

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_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

<i>digits</i>	int
<i>delta</i>	float
<i>check_list</i>	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 strong 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

13/D

14/D

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

From Pin	To Pin	Clock	Level
11/G	12/D	C1	positive
11/G	13/D	C1	positive

SEE ALSO

`create_clock(2)`
`report_constraint(2)`
`report_timing(2)`
`set_clock_groups(2)`
`set_case_analysis(2)`

`check_timing`

```
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)
```

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.

compare_interface_timing

-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

compare_interface_timing

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
```

```
...
```

```
*****
```

From	To	Type	Arc	Arc Value		
			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.spf, by inserting zero capacitances and resistances.

```
pt_shell> read_parasitics speffile.spf
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_net

Connects a net to specified pins or ports.

SYNTAX

```
int connect_net
net
object_spec
```

Data Types

<i>net</i>	string
<i>object_spec</i>	list

ARGUMENTS

net string
Specifies the name of the net to which the pins and ports are to be connected.

object_spec list
Specifies a list of pins or ports to connect to *net*.

DESCRIPTION

Connects pins and ports to a *net* in the current design. Like all other netlist editing commands, for the **connect_net** command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The **connect_net** command returns a 1 if successful and a 0 if unsuccessful.

The *net* and each pin and port specified in the *object_spec* must be in scope; that is, at or below the current instance.

There are two rules for the **connect_net** command:

- You cannot connect *net* to a pin that is already connected.
- You cannot connect across a hierarchical boundary. For example, you cannot connect a port to a net at a hierarchical level other than the top of the design.

EXAMPLES

The following example creates a cell and a net and then connects the new net to the output of the new cell:

```
pt_shell> create_net new_net1
```

```
connect_net
```

```
Information: Created net 'new_net1' in design 'top'. (NED-016)
1
```

```
pt_shell> create_cell u1 class/AN2
Information: Created cell 'u1' in design 'top'. (NED-014)
1
```

```
pt_shell> connect_net new_net1 u1/z
Information: Connected 'u1/Z' to 'new_net1'. (NED-018)
1
```

SEE ALSO

`create_cell(2)`
`create_net(2)`
`disconnect_net(2)`
`size_cell(2)`
`swap_cell(2)`
`write_changes(2)`

connect_power_domain

Connects power net information for the specified power domains.

SYNTAX

```
int connect_power_domain
power_domains
[-primary_power_net power_net]
[-primary_ground_net ground_net]
[-backup_power_net power_net]
[-backup_ground_net ground_net]
[-internal_power_net internal_power_net]
[-internal_ground_net internal_ground_net]
```

Data Types

<i>power_domains</i>	collection
<i>power_net</i>	string
<i>ground_net</i>	string
<i>internal_ground_net</i>	string
<i>internal_ground_net</i>	string

ARGUMENTS

power_domains
Specifies the power domains for which to make the power net information connections.

-primary_power_net power_net
Specifies the primary power net information for the power domains.

-primary_ground_net ground_net
Specifies the primary ground net information for the power domains.

-backup_power_net power_net
Specifies the backup power net information for the power domains.

-backup_ground_net ground_net
Specifies the backup ground net information for the power domains.

-internal_power_net internal_power_net
Specifies the internal power net information for the power domains.

-internal_ground_net internal_ground_net
Specifies the internal ground net information for the power domains.

DESCRIPTION

The **connect_power_domain** command creates logical power connections for the specified power domains. All cells in the power domain inherit the power connections. You can override the inherited power connections for a cell by using the

connect_power_net_info command.

The primary power and ground nets are used to define the main power connections for the power domain.

The backup power and ground nets are used to define power connections for always-on logic, retention registers, isolation cells, and enable-level shifter cells.

The internal power and ground nets are used to connect power nets to the switching cells present inside the power domain.

EXAMPLE

The following example connects power net information to the TOP_DOMAIN and SUB_DOMAIN power domains.

```
prompt> connect_power_domain TOP_DOMAIN \
      -primary_power_net T_VDD \
      -primary_ground_net T_VSS
1
prompt> connect_power_domain SUB_DOMAIN \
      -primary_power_net A_VDD \
      -primary_ground_net A_VSS \
      -backup_power_net VDD_Backup \
      -backup_ground_net VSS_Backup
1
```

SEE ALSO

[create_power_domain](#) (2)
[get_power_domains](#) (2)
[report_power_domain](#) (2)
[connect_power_net_info](#) (2)

connect_power_net_info

Connects the specified power net information to the specified power pins.

SYNTAX

```
int connect_power_net_info  
object_list  
-power_pin_name power_pin_name  
-power_net_name power_net_name
```

Data Types

<i>object_list</i>	list
<i>power_pin_name</i>	string
<i>power_net_name</i>	string

ARGUMENT

object_list
Specifies the leaf cells for which the power net information connections are made.

-*power_pin_name* *power_pin_name*
Specifies the name of the power pin.

-*power_net_name* *power_net_name*
Specifies the name of the power net.

DESCRIPTION

The **connect_power_net_info** command makes power net information connections for a specific power pin of a leaf cell. The pin-level connection overrides the domain-level connections made with the **connect_power_domain** command.

See the **report_power_pin_info** man page for more information.

EXAMPLE

The following example connects the power net information for the PWR pin of the PD0_INST/IO cell.

```
prompt> connect_power_net_info PD0_INST/IO \  
-power_pin_name PWR -power_net_name exp_VDD  
1
```

SEE ALSO

`connect_power_domain(2)`
`report_power_pin_info(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 for UPF mode

```
status connect_supply_net
```

```
[-ports list]  
supply_net_name
```

Data Types

<i>list</i>	list
<i>supply_net_name</i>	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 (non-hierarchical) name. The net must exist in the current scope. This  
argument is required.
```

DESCRIPTION for UPF Mode

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)
```

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

The **copy_collection** command is an efficient mechanism for creating a duplicate of an existing collection. It is more efficient and almost always sufficient to simply have more than one variable referencing the same collection. For example, if you create a collection of ports in PrimeTime and save a reference to it in a **c1** variable, assigning the value of the **c1** variable to another **c2** variable creates a second reference to the same collection:

```
pt_shell> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
pt_shell> set c2 $c1
{"U1", "U10", "U11", "U12"}
```

This has not copied the collection. There are now two references to the same collection. If you change the *c1* variable, the *c2* variable continues to reference the same collection:

```
pt_shell> set c1 [get_cells "block1"]
{"block1"}
pt_shell> query_objects $c2
{"U1", "U10", "U11", "U12"}
```

There might be instances when you really do need a copy. In those cases, the **copy_collection** command is used to create a new collection that is a duplicate of

the original.

EXAMPLES

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different than having multiple references to the same collection.

```
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\(2\)](#)

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 a detailed description of how to format a floating point argument, see the **print** UNIX man page.

```
pt_shell> cputime "%g"
3.74
```

SEE ALSO

mem(2)

create_cell

Creates cells in the current design.

SYNTAX

```
int create_cell  
[-libraries lib_spec]  
[-exact]  
cell_list  
lib_cell
```

Data Types

<i>lib_spec</i>	string
<i>cell_list</i>	list
<i>lib_cell</i>	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 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.

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

<i>period_value</i>	float
<i>clock_name</i>	string
<i>edge_list</i>	list
<i>source_objects</i>	list
<i>comment_string</i>	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]
[-edge_shift edge_shift_list]
[-add]
[-master_clock clock]
[-pll_output output_pin]
[-pll_feedback feedback_pin]
[-comment comment_string]
source_objects
```

Data Types

<i>clock_name</i>	string
<i>master_pin</i>	list
<i>divide_factor</i>	int
<i>multiply_factor</i>	int
<i>edge_list</i>	list
<i>percent</i>	float
<i>edge_shift_list</i>	list
<i>clock</i>	string
<i>output_pin</i>	list
<i>feedback_pin</i>	list
<i>comment_string</i>	string
<i>source_objects</i>	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. 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.

-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).

-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.

create_generated_clock

```

-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 } is 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_histogram

Creates a custom histogram graph.

SYNTAX

```
int create_histogram

    [-numbins number_of_bins]
    [-min min_value]
    [-max max_value]
    [-greater_than gt_value]
    [-less_than lt_value]
    [-midpoint mid_value]
    [-worst worst_side]
    [-value_label value_label]
    [-object_name object_name]
    [-title title_label]
    [- xlabel x_axis_label]
    [- ylabel y_axis_label]
    [-width window_width]
    [-height window_height]
    -attribute attribute_name
    -tcl_cmd tcl_command
    -collection object_list

int number_of_bins
float min_value
float max_value
float gt_value
float lt_value
float mid_value
string worst_side
string value_label
string object_name
string title_label
string x_axis_label
string y_axis_label
int window_width
int window_height
string attribute_name
string tcl_command
string object_list
```

ARGUMENTS

-numbins *number_of_bins*

An integer in the range of 1 to 300, that specifies the number of bins to use in creating the histogram. The default is 8.

-min *min_value*

Specifies the minimum value to use in binning objects; the default is the smallest value used in binning the objects, or the value of **-greater_than**, if specified. Any object whose associated value is less than this value is

to be dropped from the histogram with a warning message.

-max *max_value*

Specifies the maximum value to use in binning objects; the default is the largest value used in binning the objects, or the value of **-less_than**, if specified. Any object whose associated value is greater than this value is to be dropped from the histogram with a warning message.

-greater_than *gt_value*

Specifies the non-inclusive least value limit to use in binning objects; by default, there is no filtering of data on the min end of the value range. If **-min** is specified and **-greater_than** is not, the value given for **-min** is used. Any object whose associated value is less than or equal to this value is to be dropped from the histogram with a warning message.

-less_than *gt_value*

Specifies the non-inclusive greatest value limit to use in binning objects; by default, there is no filtering of data on the max end of the value range. If **-max** is specified and **-less_than** is not, the value given for **-max** is used. Any object whose associated value is greater than or equal to this value is to be dropped from the histogram with a warning message.

-midpoint *mid_value*

Specifies the midpoint value to use in interpreting histogram bins; by default, no midpoint is used. An x-axis tick mark is placed on the midpoint value and all bins to the **worst** side of the midpoint are colored red while all bins to the other side of the midpoint are colored green. The **worst** side is **none** by default but can be set to **left** or **right** using the **-worst** option.

-worst *worst_side*

Specifies whether values less than or greater than the midpoint are to be considered "worst." Allowed values are **left**, meaning that values less than midpoint are worse; **right**, meaning that values greater than the midpoint are worse; and **none** (the default), meaning that neither is worse.

-value_label *value_label*

Specifies the label to place at the head of the value column in the list view accompanying the histogram. If not specified, the default is the attribute name specified for the **-attribute** option, or the Tcl command string specified for the **-tcl_cmd** option, whichever is used. This label may also be used to construct the default title and X axis labels.

-object_name *object_name*

Specifies the label to place at the head of the object name column in the list view accompanying the histogram. If not specified, the default is the standard type name for the type of objects found in the object list supplied for the **-collection** argument. For example, if the list consists of pins, the label "Pin" is placed at the head of the object name column. If the object list consists of multiple types of objects, the label "Object" is used. This label may also be used to construct the default title and Y axis labels.

-title *title_label*

Specifies the label to place at the top of the histogram graph. If not specified, a default title is constructed by concatenating the *object_name* label and the *value_label* label.

```

-xlabel x_axis_label
    Specifies the label to place along the X axis of the histogram graph. If not
    specified, the default X axis label is the value_label label.

-ylabel y_axis_label
    Specifies the label to place along the Y axis of the histogram graph. If not
    specified, the default Y axis label is constructed with the string "Number
    of object_names" where object_name is replaced with the object_name label.

-width window_width
    Specifies the width of the window in number of pixels. The default is the
    width of the last histogram window closed when PrimeTime was last run.

-height window_height
    Specifies the height of the window in number of pixels. The default is the
    height of the last histogram window closed when PrimeTime was last run.

-attribute attribute_name
    Specifies the name of the attribute to use in constructing the histogram. The
    value of the named attribute is fetched for each object in the object list
    specified by the -collection argument. These values are then used to bin the
    objects to construct the histogram. If the attribute value is not defined for
    any object in the list, a warning message is issued and the object is excluded
    from the histogram. You must specify either -attribute or -tcl_cmd, but not
    both.

-tcl_cmd tcl_command_string
    Specifies the Tcl command to use in constructing the histogram. The Tcl
    command is evaluated for each object in the object list specified by the -
    collection argument. These values are then used to bin the objects to
    construct the histogram. If the Tcl command evaluation fails for any object
    in the list, a warning message is issued and the object is excluded from the
    histogram. You must specify either -attribute or -tcl_cmd, but not both.
    For a description of how the object in the object list is passed to the Tcl
    command, see the DESCRIPTION section.

object_list
    Specifies the collection containing the objects that will be used to generate
    the histogram.

```

DESCRIPTION

The **create_histogram** command constructs a histogram by binning the objects in the given object list using values evaluated with the given attribute name or Tcl command. You can issue **create_histogram** only while the GUI is active. Type the command into the command line field at the bottom of the GUI console window.

The object in the object list is passed to the Tcl command using the following conventions.

For each object the following is appended to the supplied Tcl command string: "-collection collection_name -index index_of_object" where collection_name is replaced by the name of the list given to the **create_histogram** command and the index_of_object is replaced with the zero-based index of the current object under

evaluation in that list.

If the object is not a timing path, timing arc, or library timing arc, the following is further appended after the **-collection** and **-index** arguments: "-object *object*" where *object* is replaced with the name of a one-object collection that contains the current object under evaluation from the object list.

Note that you cannot use the **index_collection** command as the **-tcl_cmd** argument for lists of timing paths, timing arcs, library timing arcs, and timing points.

The histogram is graphed in a histogram view accompanied by a two-column table view that displays members of a selected bar. You can select a bar in the histogram by clicking the left mouse button over the bar. When a bar is selected, the accompanying table view displays the objects in the bar. The left column displays the object names, while the right column displays the values associated with the objects used to created the histogram.

EXAMPLES

These examples show creation of custom histograms using the **create_histogram** command. All command strings must be entered in the command line field at the bottom of the GUI console window.

The following example shows creation of a histogram using the **-attribute** flag to retrieve an attribute value for each object in a collection. In this example, the **fanout_load** attribute value is used to bin all pins in the design. The graph is labeled with the title string "Pin Fanout" using the **-title** option. A new window frame is created for the new histogram by default because no view is given with a **-view** option. The new histogram view is named **view1** using the **-name** option,

```
pt_shell> create_histogram -name view1 -attribute fanout_load \
           -title "Pin Fanout" -collection [get_pins *]
```

1

The following example shows creation of another histogram binning cells by area, with the graph reusing the histogram view named **view1** created in the previous example. The histogram will have the default title "Cell area", default X axis label "area" and Y axis label "Number of cells".

```
pt_shell> create_histogram -view view1 -attribute area \
           -collection [get_cells *]
```

1

The following example shows creation of a histograms using the **-tcl_cmd** flag to use a user defined-Tcl procedure to retrieve a value for each object in a collection. First, a Tcl script file named "myscript.tcl" containing the following lines is created to define a Tcl procedure named "myproc" that retrieves the **fanout_load** attribute value for the object that is passed in the procedure arguments:

```
proc myproc {flag1 clct flag2 idx flag3 obj} {
    set result [get_attribute $obj fanout_load]
    return $result
}
```

The disregarded strings flag1, flag2, and flag3 are placeholders for the **-collection**, **-index**, and **-object** option flags, respectively, preceding the three arguments.

You can identify the object by directly using the object in the third argument as in the myproc procedure above, or by using the collection name and the object index in the first and second arguments, respectively. The following procedure, myproc2, performs the same function as myproc but retrieves its objects using the first and second arguments:

```
proc myproc2 {flag1 clct flag2 idx flag3 obj} {
    set mypin [index_collection $clct $idx]
    set result [get_attribute $mypin fanout_load]
    return $result
}
```

You can now source the script file to define the procedures within the PrimeTime Tcl interpreter context:

```
pt_shell> source myscript.tcl
1
```

Once the procedures have been defined, you can reference them from the **-tcl_cmd** argument in a **create_histogram** command invocation as in the following example.

```
pt_shell> create_histogram -name view1 -tcl_cmd myproc \
           -title "Pin Fanout" -collection [get_pins *]
1
```

In this example, the embedded command **[get_pins *]** is first interpreted to create a collection of all pins in the design. The Tcl procedure "myproc" is then invoked for each pin object in the collection to retrieve a value per pin to use in creating the histogram.

SEE ALSO

collections (2).

create_ilm

Extracts interface logic model and writes it to a new directory. Also, sets the **is_interface_logic_pin** attribute on pins of the current design that are part of its interface logic model.

SYNTAX

```
int create_ilm
[-script_format format]
[-instances instance_list]
[-ignore_ports port_list]
[-auto_ignore]
[-verbose]
[-ignore_boundary_pins pin_list]
[-include incl_list]
[-verification_script]
[-latch_level levels]
[-context_borrow]
[-keep_ignored_fanout]
[-include_pins pin_list]
[-critical_pins]
[-traverse_disabled_arcs]
[-parasitics_options para_options]
[-sdf_options sdf_options]
[-block_scope]
[-block_scope_only]
[-scope_scenario scenario_name]
[-validate valid_list]
```

Data Types

<i>format</i>	string
<i>instance_list</i>	list
<i>port_list</i>	list
<i>pin_list</i>	list
<i>incl_list</i>	list
<i>levels</i>	int
<i>para_options</i>	list
<i>sdf_options</i>	list
<i>scenario_name</i>	string

ARGUMENTS

```
-script_format format
    Specifies the output format for the script. Allowed values are ptsh (the
    default) for pt_shell, dcs for dc_shell, and dctcl for dc_shell-t.

-instances instance_list
    Specifies the list of instances for which ILMs need to be generated. Separate
    ILMs are generated for each instance and written out to directories named
    after the instance name.
```

-ignore_ports *port_list*

Specifies a list of input or output ports whose fanout or fanin is to be ignored when setting the **is_interface_logic_pin** attribute. 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 impacting 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**, **-ignore_boundary_pins**, and **-auto_ignore** options are mutually exclusive.

-auto_ignore

Enables automatic determination of ports whose fanout should be ignored when setting 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 (default 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**, **-ignore_boundary_pins** and **-auto_ignore** options are mutually exclusive.

-verbose

Indicates that information about the number of cells and nets in the original design and in the ILM netlist is to be written to standard output.

-ignore_boundary_pins *pin_list*

Specifies a list of boundary pins whose fanout or fanin is to be ignored when placing the **is_interface_logic_pin** attribute while extracting an instance ILM. Works similar to the **-ignore_ports** option but works for instance ILMs. This option can only be used along with the **-instances** option.

The **-ignore_ports**, **-ignore_boundary_pins** and **-auto_ignore** options are mutually exclusive.

-include *incl_list*

Specifies the additional categories to be included in the ILM. Allowed values are **net_pins**, **boundary_cells**, **si_delay_pins** and **si_noise_pins**. The **net_pins** argument 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 non-SDF flows and delay calculation is performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net. The **net_pins** does not affect unpropagated clock nets; that is, nets in the clock network that have user-specified source latency.

The **boundary_cells** argument specifies that the ILM has to include the boundary cells for all input ports. By default, the boundary cells would be present for all inputs except for the ignored ports. This option allows users to include boundary cells for the ignored input ports also so that DRC checks can be performed at top level.

The **si_delay_pins** argument specifies that the ILM has to include additional logic required to perform crosstalk delay analysis at the chip-level. Since the crosstalk flow involves detailed parasitics, the **si_delay_pins** argument also turns the **net_pins** argument on.

The **si_noise_pins** argument specifies that the ILM has to include additional logic required to perform noise analysis at the chip-level. Since the noise flow involves detailed parasitics, the **si_noise_pins** argument also turns the **net_pins** argument on.

-verification_script

Specifies that a script needs to be generated that can be used for the verification of ILM. By default, **create_ilm** command generates a script that can be used when the ILM is instantiated at upper level of hierarchy. This option additionally generates a script that can be used to validate the ILM against the original block from which the ILM was generated.

-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 the *levels* option. 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.

When you use this option, note that the value of the *levels* argument 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* + 1th 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. So, 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. 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 the `pin_list` option. 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. Use this option to define additional interface logic pins that are otherwise removed in the model. There is not 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.

The `-include {si_delay_pins}` option is not currently supported for critical pins ILM.

`-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.

`-parasitics_options para_list`

Specifies that the parasitics need to be generated for ILM. Allowed values for the list are: `spef_format`, `sbpf_format`, `input_port_nets` and `constant_nets`.

The `spef_format` argument specifies that the parasitics are to be written in SPEF.

The `sbpf_format` argument specifies that the parasitics are to be written in SBPF.

The `input_port_nets` argument indicates that parasitic information is to be written out for nets connected to the input ports on the current design. By default, this information is not written out.

The `constant_nets` argument indicates that parasitic information is to be written out for constant nets also. By default, this information is not written out.

When you use the `-parasitics_options` option, all net pins of ILM nets are included. In other words, the `-include {net_pins}` option is always assumed when you use the `-parasitics_options` option.

`-sdf_options sdf_options`

Specifies that SDF needs to be generated for ILM. Allowed values for the list are: `annotated`, `no_edge`, `2.1_version`, `3.0_version`, `input_port_nets`, `output_port_nets`.

The `annotated` argument 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` commands.

The `no_edge` argument indicates that the generated SDF is not to include any edges (posedge or negedge) for combinational I/O paths.

The `2.1_version` argument selects SDF version 2.1. This is the default.

The **3.0_version** argument selects SDF version 3.0.
The **input_port_nets** argument 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.
The **output_port_nets** argument 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.

-block_scope

Specifies that the block scope information needs to be captured for the timing model. If specified, this creates a separate file that contains scope information for the block that is replaced by ILM at the top-level analysis. Note that the scope information that is captured and stored is not impacted by the **hier_scope_check_defaults** variable.

-block_scope_only

Specifies that only the block scope information needs to be captured, and not the ILM. This option is useful for the later runs when an ILM was already generated but the scope of block-level validation has changed. If specified, this creates a separate file that contains scope information for the block that is replaced by ILM at the top-level analysis. Note that the scope information that is captured and stored is not impacted by the **hier_scope_check_defaults** variable.

-scope_scenario scenario_name

Specifies the name of the scenario for which the scope data needs to be labeled and later checked for. The scope information captured and stored in the scope data file is marked as corresponding to the given scenario. If not specified, a fixed default name is used internally.

-validate valid_list

Specifies what should be validated after the model is created. The allowed value is **timing**.

DESCRIPTION

Extracts an interface timing model (ILM) for the design or given list of instances. The ILM is written to a directory with name of the design, or the hierarchical name of instance with forward slash (/) replaced by underscore (_). The location of this potentially new directory is given by the **pt_ilm_dir** variable.

The command generates the following files:

- *dir_name/ilm.v* (Verilog design for the ILM)
- *dir_name/ilm_inst.pt.gz* (script to apply the instance constraints)

Additionally, the command might generate:

- *dir_name*/ilm.{sbpf, spef.gz} (Parasitics if **-parasitics_options** is given)
- *dir_name*/ilm.sdf (SDF file if **-sdf_options** is given)
- *dir_name*/ilm_verif.pt.gz (verification script for the design)
- *dir_name*/ilm.txt (list of aggressor annotation pins if **-include {si_delay_pins}** is given)
- <*dir_name*>/ilm.scope (scope information for the block being replaced by model)

Also, the command sets the **is_interface_logic_pin** attribute on pins of the current design that are part of its interface logic.

The interface logic on a block contains all cells whose timing is impacted by, or impacts, the external environment of a block. The following list describes such parts of 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.

You can include additional logic by using the **-include_pins** option. If the **-include {si_delay_pins}** option is given, additional register-to-register logic can be pulled in that is needed for crosstalk analysis at chip level.

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.

Also, you can use the command to generate scope information for the block that is replaced with the ILM at top-level.

If the **-validate** option is given, and the **hier_modeling_version** variable is set to 2.0, the model is validated automatically after the model is created.

EXAMPLES

The following example extracts ILM and sets 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> create_ilm -ignore_ports [get_ports {port1 port2 port3}]
```

The following example extracts ILM for instance I1, additionally includes three levels of logic from internal pin I1/ff/Q and also includes internal crosstalk pins needed for top-level crosstalk analysis.

```
pt_shell> set ilm_pins [all_fanout -level 3 \
-flat [get_pin I1/ff/Q]]
```

```
pt_shell> create_ilm -instances {I1} \
-include_pins $ilm_pins -include {si_delay_pins}
```

The following example extracts a critical pins interface timing model.

```
pt_shell> create_ilm -critical_pins -include {net_pins}
```

The following example extracts ILM by placing 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. Also, it creates scope information for the block.

```
pt_shell> printvar ilm_ignore_percentage
ilm_ignore_percentage = "35"
```

```
pt_shell> create_ilm -auto_ignore -latch_level 1 -block_scope
```

SEE ALSO

```
check_block_scope(2)
hier_modeling_version(3)
hier_scope_check_defaults(3)
ilm_ignore_percentage(3)
pt_ilm_dir(3)
```

create_merged_modes

Merges modes from the defined scenarios so that the timing scenarios are reduced.

SYNTAX

```
status create_merged_modes
[-test_only]
[-interactive]
[-keep_all_clocks]
[-match_clock_names]
[-value_tolerance tolerance]
```

Data Types

```
tolerance      float
method
```

ARGUMENTS

```
-test_only
    Prints the mode merging report on which modes are merged and why certain modes
    are not be merged but does not perform the actual merging.

-interactive
    Determine which modes are merged and why certain modes are not merged by doing
    -test_only analysis, open the report in a GUI for interactive debugging.

-keep_all_clocks
    Keeps all clocks separately. If you do not specify this option, mode merging
    adds only one clock into the merged mode when identical clocks (with the same
    waveform and sources) are present in more than one individual modes.

-match_clock_names
    Specifies that clock names should also be matched in addition to the waveform
    and sources. If you do not specify this option, mode merging adds only one
    clock into the merged mode when identical clocks (with the same waveform and
    sources) are present in more than one individual mode.

-value_tolerance tolerance
    When two modes have constraint values (for example, of set_clock_uncertainty,
    set_input_delay etc) that differ than certain percentage, the two modes are
    not merged. By default, a tolerance value of 0.01 is used which means that
    if the values are different by more than 1 percent, then the modes are not
    merged. You can use this option to change the tolerance value, thereby
    allowing for tradeoff between level of mode reduction and pessimism. A higher
    (lower) tolerance value leads to more (less) reduction but more (less)
    pessimism in merged constraints.
```

DESCRIPTION

The **create_merged_modes** command merges modes from the defined scenarios so that the

timing scenarios are reduced. Therefore, this command is only available in multi-scenario analysis. For mode merging, the scenarios must have been created previously using **-mode** and **-corner** options of **create_scenario**.

After the modes are merged, the merged modes should be used with the same **-corner** option as was used for individual modes. Merged mode constraints are created under *multi_scenario_working_directory* with name **merged_constraints**. Constraints are also written for unmerged modes in the directory. In addition, details of mode merging results including why certain modes could not be merged are written to text file by name **mode_merging_report.txt** in the **merged_constraints** directory.

Next, you describe how various commands are dealt with for mode merging. Mode merging merges pure mode commands (such as **create_clock**) and scenario (combination of mode and corner) commands. For scenario commands (such as **set_input_delay**), it is assumed that the command is present in all corners of a given mode but the values can differ in each mode. For pure corner commands, it assumes that the command is present in all modes of the same corner.

Mode merging handles the following pure mode and scenario commands:

- **create_clock**
- **create_generated_clock**
- **group_path**
- **set_annotated_transition**
- **set_case_analysis**
- **set_clock_gating_check**
- **set_clock_groups**
- **set_clock_latency**
- **set_clock_sense**
- **set_clock_transition**
- **set_clock_uncertainty**
- **set_disable_timing**
- **set_drive_resistance**
- **set_driving_cell**
- **set_false_path**
- **set_fanout_load**
- **set_ideal_latency**
- **set_ideal_network**
- **set_ideal_transition**
- **set_input_delay**
- **set_input_transition**
- **set_load**
- **set_max_capacitance**
- **set_max_delay**
- **set_max_fanout**
- **set_max_time_borrow**
- **set_max_transition**
- **set_min_delay**
- **set_mode**
- **set_multicycle_path**
- **set_output_delay**
- **set_propagated_clock**
- **set_sense**

Although mode merging does not merge the following pure corner commands, the tool captures and writes these commands to merged mode scripts:

- create_operating_conditions
- set_annotated_check
- set_annotated_clock_network_power
- set_annotated_delay
- set_annotated_power
- set_aocvm_coefficient
- set_aocvm_table_group
- set_connection_class
- set_coupling_separation
- set_cross_voltage_domain_analysis_guardband
- set_domain_supply_net
- set_dont_override
- set_dont_touch
- set_dont_touch_network
- set_equal
- set_input_noise
- set_isolation
- set_isolation_control
- set_level_shifter_strategy
- set_level_shifter_threshold
- set_lib_rail_connection
- set_library_driver_waveform
- set_max_area
- set_min_capacitance
- set_min_pulse_width
- set_noise_derate
- set_noise_immunity_curve
- set_noise_lib_pin
- set_noise_margin
- set_noise_parameters
- set_operating_conditions
- set_opposite
- set_port_fanout_number
- set_power_derate
- set_rail_voltage
- set_related_supply_net
- set_resistance
- set_retention
- set_retention_control
- set_setup_hold_pessimism_reduction
- set_si_aggressor_exclusion
- set_si_delay_analysis
- set_si_noise_analysis
- set_si_noise_disable_statistical
- set_steady_state_resistance
- set_supply_net_probability
- set_switching_activity
- set_temperature
- set_timing_derate
- set_voltage
- set_wire_load_min_block_size
- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group

The following list shows scenario commands that might need user intervention. Check these commands for correctness. For example, some clock names might need to be changed based on MMODE-008 messages.

- set_power_clock_scaling
- set_pulse_clock_max_transition
- set_pulse_clock_max_width
- set_pulse_clock_min_transition
- set_pulse_clock_min_width

Currently, the following commands are not supported in mode merging; if constraints contain these commands, mode merging issues an error message:

- read_ddc
- read_milkyway
- set_active_clocks

EXAMPLES

In the following example, an iterative loop creates a set of scenarios. The same single.tcl script is used for all scenarios but the corner variables distinguishes the scenarios. Finally, the modes are merged using create_merged_modes command. Modes M1 and M2 are merged to new mode merged_1. Mode M3 is not merged.

```
pt_shell> foreach corner {bc tc wc} {
    foreach mode {M1 M2 M3} {
        create_scenario
        -mode ${mode}
        -corner ${corner}
        -common_variables {mode corner}
        -common_data "single.tcl"
    }
}

pt_shell> create_merged_modes
...
          Merged Mode Generation Summary
Original Modes           Merged Mode     Corners
-----
M1, M2                   merged_1      bc, tc, wc
M3                      M3            bc, tc, wc
```

SEE ALSO

[create_scenario\(2\)](#)

create_net

Creates nets in the current design.

SYNTAX

```
int 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 the section entitled "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.

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

<i>name</i>	string
<i>library_name</i>	string
<i>process_value</i>	float
<i>temperature_value</i>	float
<i>voltage_value</i>	float
<i>tree_type</i>	string
<i>calc_mode</i>	string
<i>rail_value_pairs</i>	Tcl 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_placement_blockage

Creates a new placement blockage.

SYNTAX

```
status create_placement_blockage
-bbox {llx1 lly1 urx1 ury1}
[-name blockage_name]
```

Data Types

llx1	float
lly1	float
urx1	float
ury1	float
blockage_name	string

ARGUMENTS

```
-bbox {llx1 lly1 urx1 ury1}
      Specifies the coordinates (in microns) of the bounding box of the placement
      blockage.

-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_domain

Creates a power domain at the specified scope, which provides a power supply distribution network.

SYNTAX

```
string create_power_domain
domain_name
[-elements element_list]
[-include_scope]
[-scope instance_name]
[-supply {supply_set_handle [supply_set_ref]}]
[-update]
```

Data Types

<i>domain_name</i>	string
<i>element_list</i>	list
<i>instance_name</i>	string
<i>supply_set_handle</i>	string
<i>supply_set_ref</i>	string

ARGUMENTS

domain_name

Specify the name of the power domain to be created. The name should be a simple (non-hierarchical) name. You must specify this argument.

If there is a power domain with the same name in the specified scope, the power domain cannot be created. If there is a hierarchical/leaf cell instance or port with the same name in the specified scope, the power domain cannot be created either.

-elements element_list

Specify a list of cells which are added as the extent of the power domain. The list of cells should be hierarchical, macro, or I/O pad cells, but PrimeTime accepts any cell. Specified cells cannot be added in other power domains of the same scope.

If neither **-elements** nor **-include_scope** is specified, the power domain consists of the current scope and any of its children not specified as elements in another **create_power_domain** command.

-include_scope

If this option is specified, the scope of the power domain is also included in the extent of the power domain. This means all elements within the current scope get the supply as the power domain.

-scope instance_name

Specify in which scope the power_domain is going to be created. The instance name is the name of a hierarchical cell.

If this option is not specified, the power domain is created in the current scope.

-supply {*supply_set_handle* [*supply_set_ref*]}

Specifies the *supply_set_handle* for the domain. If *supply_set_ref* is also specified, the domain *supply_set_handle* is associated with the specified *supply_set_ref*. This option can be specified multiple times.

Either predefined or user-defined handle names can be used. The predefined *supply_set_handles* are:

- **primary**: primary supply set of this power domain
- **default_isolation**: default isolation supply set for any isolation strategy applied to this power domain if the isolation power or ground is not specified in the strategy
- **default_retention**: default retention supply set for any retention strategy applied to this power domain if the retention power or ground is not specified in the strategy

Handles of the form **extra_supplies_[0-9]+** are accepted but ignored in PrimeTime.

-update

Updates the supply set association of an existing power domain.

The **-update** option is used along with and only with the *domain_name* and **-supply** options. It is an error to use the **-update** option with any other option.

This option allows you to

- Add a new supply set handle name without explicitly defining a supply set association
- Add a supply set to be associated with a previously declared supply set handle
- Add a new supply set handle and a supply set association in a single command

• Add cells to a previously created domain.

NOTE: It is an error to update a supply set handle that you have already associated with a supply set. It is an error to add cells that are already root cells of the domain.

DESCRIPTION

The **create_power_domain** command enables you to create a power domain in the specified scope. A power_domain is a collection of design elements that share a primary power and ground power net. The logic hierarchy level where a power domain is created is called the scope of the power domain. Any design elements that belong to a power domain are said to be in the extent of that power domain. While a design element can be in the scope of a number of power domains, it can only be in the extent of one power domain.

A power domain could have several supply_nets, supply_nets are connected to power_domain via supply_port. And a power_switch of a power_domain could be used to turn on/off the power supply of part/whole power domain. The supply net, supply port, and power switch can be created using the **create_supply_net**, **create_supply_port** and **create_power_switch** commands respectively.

This command returns collection object (the newly created power domain). It returns null string if it failed.

Non-UPF Mode (Power Domains Mode): This command works in power domains mode as well, and take following options instead.

string **create_power_domain**

domain_name

[**-power_down**]

[**-power_down_ctrl list**]

[**-power_down_ack list**]

[**-power_down_ctrl_sense <0 or 1>**]

[**-object_list list**]

-power_down

Use this to specify the new power domain is a power down domain. This option is a prerequisite for the **-power_down_ctrl** option.

-power_down_ctrl list

Use this to specify the power down control net for the new power domain. The **-power_down** option is required in order to use this option. It is also a prerequisite for the **-power_down_ack** option.

-power_down_ack list

Use this to specify the power down acknowledge net for the new power domain. The **-power_down_ctrl** option is required in order to use this option.

-power_down_ctrl_sense <0 or 1>

Use this to specify the sense of the power down control net for the new power domain. It take values of 0 or 1. It specifies the active low or active high sense type for power control signal. The **-power_down_ctrl** option is required in order to use this option.

-object_list list

A list of hierarchical cells that are associated with the new power domain. When this option is absent, the new power domain is assumed to be the top level domain. For each design, there can only be one top-level domain. Also, each hierarchical cell can only be associated with one power domain.

The **create_power_domain** command creates a new power domain for the current design. The power domains for each design must be uniquely named. There can be only one design-wise top level power domain.

Use **-power_down** to specify a new domain is a power down domain. An always-on power domain is created without this switch. For a power down domain, use **-power_down_ctrl** to specify an optional power down control net; use **-power_down_ack** to specify an optional power down acknowledge net.

Use **-object_list** to associate a list of hierarchical cells with the new power domain. When this option is not used, the new power domain is considered as top-level power domain. Each hierarchical cell can only belong to one power domain.

By default, power down control and power acknowledge signals are marked dont touch. Set the **dont_touch_power_domain_control_nets** variable to false if you do not want these signal marked dont_touch.

EXAMPLES

UPF Mode:

The following example creates two power_domains in scope INST1:

```
pt_shell> create_power_domain PD1 -elements INST1/SUB_INST -scope INST1
{ "INST1/PD1" }

pt_shell> create_power_domain PD2 -elements INST1/SUB_INST -scope INST1
Error: Cell 'INST1/SUB_INST' is already in the extent of power domain 'PD1'. (UPF-001)

pt_shell> create_power_domain PD2 -scope INST1 -include_scope
{ "INST1/PD2" }
```

The following example creates a power domain in scope INST1, and uses a supply set to provide the default primary power and ground, and a different supply set to provide the default retention power and ground nets.

```
pt_shell> create_power_domain PD3 -scope INST1 -supply {primary set1} \
-supply {default_retention set2}
```

The following example updates an existing power domain with the **-supply** option.

```
prompt> create_power_domain PD_MID -scope mid1 -supply {abc}
mid1/PD_MID
prompt> create_power_domain PD_MID -scope mid1 -update \
-supply {primary SS1} -supply {abc SS1}
mid1/PD_MID
```

Non-UPF Mode:

The following examples use the command to create power domains.

```
pt_shell> create_power_domain TOP_DOMAIN
1

pt_shell> create_power_domain SUB_DOMAIN -power_down \
-power_down_ctrl [get_nets pd_ctrl] \
-power_down_ack [get_nets pd_ack] \
-object_list [get_cells mid1]
1
```

SEE ALSO

report_power_domain(2)
help UPF
help power domains
power_domains_compatibility(3)

create_power_group

Creates a power group of cells in the current design.

SYNTAX

```
int create_power_group
-name group_name
-default
object_list
```

Data Types

<i>group_name</i>	string
<i>object_list</i>	list

ARGUMENTS

-name *group_name*

Specifies the name for the power group. The name should not conflict with names of existing power groups.

-default

Indicates that the specified power group is predefined and the default list of cells are to be included in the power group. This option is mutual exclusive with the *object_list* option.

object_list

Specifies a list of cells to be included in this power group. This option is mutual exclusive with the -default option.

DESCRIPTION

Group a list of cells of special interests (not necessarily in the same hierarchy) to form a power group. The **update_power** and **report_power** commands are able to generate power waveform and power report for this power group.

The cells contained in power groups are leaf cells. If a hierarchical cell is specified in the *object_list* option, all the leaf cells contained by the hierarchical cell, instead of the hierarchical cell itself, are added to the power group.

There are several power groups predefined and automatically created by the tool. However, if a predefined power group has been removed, the -default option can be used to regenerate it.

The following is a list of the predefined power groups ordered by priority from high to low. Note that the predefined power groups are not supposed to be overlapping with one another. If one cell happens to belong to more than one group, it is put into the group with the higher priority.

- **io_pad** - All I/O PAD cells.
- **memory** - All memory cells.
- **black_box** - All black boxes.
- **clock_network** - Cells in the clock network. The contents is consistent with the results from the **get_clock_network_objects -type cell** command. If the **power_clock_network_include_clock_gating_network** variable is set to *true*, the contents is consistent with the results from the **get_clock_network_objects -type cell -include_clock_gating_network** command.
- **register** - The latches, flip-flops driven by the clock network. The contents is consistent with the results from the **get_clock_network_objects -type register** command. If the **power_clock_network_include_clock_gating_network** variable is set to *true*, the contents is consistent with the results from the **get_clock_network_objects -type register -include_clock_gating_network** command.
- **sequential** - All other sequential cells.
- **combinational** - All other combinational cells.

EXAMPLES

In the following example, a power group called `clock_tree` is created to accommodate all the clock tree cells.

```
pt_shell> create_power_group -name clock_tree [get_clock_network_objects -type cell]
1
```

In the following example, the predefined power group `clock_network` has been removed earlier, now you want to create it again with the default list of cells.

```
pt_shell> create_power_group -name clock_network -default
1
```

SEE ALSO

```
report_power_groups(2)
remove_power_groups(2)
get_power_group_objects(2)
update_power(2)
report_power(2)
power_clock_network_include_clock_gating_network(3)
```

create_power_net_info

Creates a power net.

SYNTAX

```
int create_power_net_info
power_net_name
-power
-gnd
[-switchable]
[-nominal_voltages nominal_voltage_list]
[-voltage_ranges voltage_range_list]
```

Data Types

<i>power_net_name</i>	string
<i>nominal_voltage_list</i>	list
<i>voltage_range_list</i>	list

ARGUMENTS

power_net_name

The name of the new power net info to be created.

-power

Specifies the type of the new power net is "power". Mutually exclusive with -gnd option.

-gnd

Specifies the type of the new power net is "gnd". Mutually exclusive with -power option.

-switchable

Specifies that this power net is switchable (can be cut off externally, or, if driven by an internal switch, cut off internally). This option can only be used with -power.

-nominal_voltages nominal_voltage_list

Specifies the list of nominal voltages this power net has been designed to operate. The actual operating voltage of the power net can deviate from the nominal within a certain tolerance as specified by *voltage_ranges_list*. This optional argument must be specified together with *voltage_ranges_list*. This option can only be used with -power.

-voltage_ranges voltage_range_list

Specifies the list of allowed voltage ranges around the nominal voltages that this power net has been designed to operate. This optional argument must be specified together with *nominal_voltages_list*. This option can only be used with -power.

DESCRIPTION

The **create_power_net_info** command creates a new power net info for the current design. The power net is either of type power or of type gnd.

EXAMPLES

The following examples use the command to create power nets.

```
prompt> create_power_net_info VSS_net -gnd
1
prompt> create_power_net_info VDD1_net -power
1
prompt> create_power_net_info VDD2_net -power \
    -switchable -nominal_voltages {0.9 1.08} \
    -voltage_ranges {0.88 0.92 1.05 1.10}
1
```

SEE ALSO

`report_power_net_info(2)`
`set_voltage(2)`

create_power_rail_mapping

Map the power rails defined in the libraries to the physical power rails existing in the design.

SYNTAX

```
string create_power_rail_mapping
design_rail
[-lib_rail_name power_rail_name]
[-cells cell_list]
[-off_condition condition]
[-default]
```

Data Types

<i>design_rail</i>	string
<i>power_rail_name</i>	string
<i>cell_list</i>	list
<i>condition</i>	string

ARGUMENTS

design_rail
Defines the power rail used in the design.

-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. Please make sure 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.

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 user 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 technology 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:

create_power_rail_mapping

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 technology 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' technology 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 V1 -cells ls_block
pt_shell> create_power_rail_mapping VDD2 -lib 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

```
report_power_rail_mapping(2)
current_power_rail(2)
update_power(2)
report_power(2)
read_vcd(2)
read_saif(2)
set_switching_activity(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

```
string 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}]
```

Data Types

switch_name	string
domain_name	string
port_name	string
supply_net_name	string
net_name	string
boolean_function	list
delay	string
state_name	string
input_supply_port	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.

```


    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.


    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 once. One power switch can have multiple
      control ports.


    Specifies the on state, the relevant input supply port, and its Boolean
    function. You can specify this option multiple times.


    Specifies the off state and its relevant Boolean function. This option is
    currently ignored by the PrimeTime tool.


    Specifies the partial state and its relevant Boolean function. This option
    is currently ignored in the PrimeTime tool.


    Specifies the error state and its relevant Boolean function. This option is
    currently ignored by the PrimeTime tool.


    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.


    Specifies the acknowledge port on the switch and the corresponding delay.
    This option is currently ignored by the PrimeTime tool.

```

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 VNII1} \
           -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

`report_power_switch(2)`
`get_power_switches(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]
-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

<i>arc_name</i>	string
<i>port_name</i>	string
<i>port_spec</i>	list
<i>name</i>	string
<i>multiplication_factor</i>	float
<i>constraint_value</i>	float

ARGUMENTS

```
-name arc_name
      Specifies the name of the constraint arc.

-setup
      Creates a setup arc.

-hold
      Creates a hold 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 constraints 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** and **-hold** options to specify either a setup or hold arc. You must use one of the switches, but you cannot use both 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 arc, the quick timing model (QTM) parameter global setup time is added to the value of the specified setup. If the constraint arc is a hold arc, the global hold time is added to the value of hold specified. You must define the global setup time before creating any setup arc and global hold time before creating any hold arc. The command checks to see if these parameters are defined. Global setup and time, and global hold 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)
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

<i>arc_name</i>	string
<i>port_spec</i>	list
<i>port_spec</i>	list
<i>path_type</i>	string
<i>multiplication_factor</i>	float
<i>delay_value</i>	float
<i>total_value</i>	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 name
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)  
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

<i>lib_cell_name</i>	string
<i>input_pin_name</i>	string
<i>output_pin_name</i>	string
<i>rtrans</i>	float
<i>ftrans</i>	float
<i>drive_type_name</i>	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)

create_qtm_drive_type

```
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

<i>master_clock_name</i>	string
<i>divide_factor</i>	int
<i>multiply_factor</i>	int
<i>generated_clock_name</i>	string

ARGUMENTS

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.

-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).

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)
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

<i>lib_cell_name</i>	string
<i>input_pin_name</i>	string
<i>load_type_name</i>	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

<i>name</i>	string
<i>pin_name</i>	string
<i>count</i>	integer
<i>path_type_name</i>	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 starting 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_path_type
148

create_qtm_port

Creates a quick timing model (QTM) port.

SYNTAX

```
string create_qtm_port
-type port_type
port_list
```

Data Types

<i>port_type</i>	string
<i>port_list</i>	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)  
get_qtm_ports(2)  
report_qtm_model(2)  
save_qtm_model(2)  
set_qtm_port_drive(2)  
set_qtm_port_load(2)
```

create_scenario

Creates a scenario for multi-scenario analysis.

SYNTAX

```
status create_scenario
  [-name scenario_name]
  [-mode mode_name]
  [-corner corner_name]
  [-common_data file_list]
  [-common_variables variable_list]
  [-specific_data file_list]
  [-specific_variables variable_list]
  [-image image]
```

Data Types

scenario_name	string
mode_name	string
corner_name	string
file_list	list
variable_list	list
image	string

ARGUMENTS

-name scenario_name

Specifies a single scenario. If you use the **-mode** and **-corner** options, PrimeTime automatically names the scenario *mode_name_corner_name*.

-mode mode_name

Specifies the mode comprising the scenario. You must use this option with the **-corner** option; do not use this option with the **-name** option. PrimeTime automatically names the scenario *mode_name_corner_name*.

-corner corner_name

Specifies the corner comprising the scenario. You must use this option with the **-mode** option; do not use this option with the **-name** option. PrimeTime automatically names the scenario *mode_name_corner_name*.

-common_data file_list

Lists files that contain commands and data that is common to more than one scenario. These files are used to create a common image for all scenarios which share the same common data. This option cannot be used in conjunction with the **-image** option.

-common_variables variable_list

Lists variables that are common to more than one scenario. These variables are used in conjunction with the common data files to determine commonality between scenarios. The value of the variable currently at the master are set on the slave before the sourcing of the common data files. This option cannot be used in conjunction with the **-image** option.

```

-specific_data file_list
    Lists files that contain commands and data that is specific to this scenario.
    This data is not used on any other scenario or in the common image generation.
    This data that is specific to the scenario is sent directly to the slave and
    executed after the common design image is generated. This option can be used
    in conjunction with the -image option.

-specific_variables variable_list
    Lists variables that are specific to this scenario. The value of the variable
    currently at the master is set on the slave before the sourcing of the
    specific data files. This option can be used in conjunction with the -image
    option.

-image image
    Allows a scenario to be created with a previously saved scenario image. The
    image can be an image generated within a standard PrimeTime session, an image
    generated with a save_session call from the DMSA master or a current_image
    generated during a DMSA session. The -specific_variables and the -
    specific_data options can be specified in conjunction with the -image option.
    The specific variables are set after the image are loaded and then the
    specific data files are executed.

```

DESCRIPTION

The **create_scenario** command creates a scenario based on the data specified by the arguments. A scenario can be created with a set of scripts and variables. When a set of scenarios shares the same data and variables, the tool creates a common design image that is shared across the set of scenarios. Alternatively, a scenario can be created with a previously saved image. This command does not actually start the analysis. The analysis starts when the first merged reporting or the **remote_execute** command is issued.

EXAMPLES

The following example creates a scenario named `scen1`, which is described by the `common_s1` and `specific_s1` files.

```
pt_shell> create_scenario -name scen1 -common_data {common_s1.pt} \
           -specific_data {specific_s1.pt}
1
```

In the following example, an iterative loop creates a set of scenarios. The same `single.tcl` script is used for all scenarios, but the `corner` variable distinguishes the scenarios.

```
foreach corner {bc tc wc} {
    create_scenario
        -name ${corner}
        -common_variables {corner}
        -common_data "single.tcl"
}
```

SEE ALSO

`current_scenario(2)`
`remove_scenario(2)`
`report_multi_scenario_design(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

```
Boolean create_si_context
[-include include_list]
[-instances instance_list]
[-parasitics_options para_options]
[-top_inst instance_name]
[-no_design_parasitics]
```

Data Types

<i>include_list</i>	list
<i>instance_list</i>	list
<i>para_options</i>	list
<i>instance_name</i>	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

<i>domain_name</i>	string
<i>supply_net_name</i>	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_supply_set

Creates a set of supplies that can be used to define the power network. A supply set is created in the current logic hierarchy.

SYNTAX

```
string create_supply_set
       supply_set_name
       [-function {function_name net_name}]*
       [-update]
```

Data Types

<i>supply_set_name</i>	string
<i>function_name</i>	string
<i>net_name</i>	string

ARGUMENTS

supply_set_name
Specifies the name of the supply set to be created. The name should be a simple (non-hierarchical) name.
If a supply set with the same name exists in the current scope, the supply set is not created.
An exception to this rule is when you use the *-update* option. In this case, the supply set name must be the same as an existing supply set in the same scope.
This argument is required.

*-function {*function_name* *net_name*}*
Specifies the functional capability to which an actual supply net should be associated.
This argument takes two parameters. The first parameter is the name of the function, which is either *power*, *ground*, *pwell* or *nwell*. The second parameter is the name of the supply net that is in the same hierarchical scope of the supply set.
pwell and *nwell* are bias functions, and they are only accessible when *design_attribute enable_bias* is true. This attribute could be set using *set_design_attribute*.
This argument is required when the *-update* argument is used. Otherwise, it is optional.

-update
Instructs the tool to associate an actual supply net to the *power*, *ground*, *pwell* and *nwell* function of an existing supply set. It is an error if the supply set specified does not exist. Only one update is allowed for each functional of the supply set. If your update is not successful, the tool issues an error message. You can continue to update the functionality, until your update is successful.
This argument is optional. When the *-update* argument is used, the *-function*

argument must also be used.

DESCRIPTION

The **create_supply_set** command creates a supply set in the current scope. A supply set eliminates the requirement of a supply net and supply ports in the design in the frontend tool. However, actual supply nets must be associated with the supply sets later on in the flow before physical synthesis is done.

A supply set supports the use of the following functions in the design:

- Power
- Ground
- Pwell
- Nwell

These functions, when unassociated with any supply net, can be used as supply nets for any purpose. They can be accessed by referencing them through the name of the supply set. To access the *power* function, you must specify *supply_set_name.power* and to access the *ground* function, you must specify *supply_set_name.ground*. Similarly, to access the *pwell* function, you must specify *supply_set_name.pwell* and to access the *nwell* function, you must specify *supply_set_name.nwell*.

The functions of a supply set can be used just as any other supply net in the UPF design, with no domain restrictions. However, domain restrictions are applicable when an *-update* is called. This means that the actual supply nets, being associated with a supply set, are checked for their definitions in the domains where the supply set functions have been used.

Calling an update on an existing supply set means replacing the supply set functions with actual supply nets. After doing an *-update*, the actual supply net can be referenced through the supply set or by its actual name.

EXAMPLES

The following example creates a supply set named *primary_sset* in the current logical hierarchy.

```
pt_shell> create_supply_set primary_sset
primary_sset
```

The following example illustrates how a supply set function can be used.

```

pt_shell> set_design_attribute -attribute enable_bias TRUE -elements { . }
Information: Resolving '.' to the current scope '<top design>'.
1

pt_shell> create_supply_set primary_sset
primary_sset

pt_shell> create_power_domain PD_TOP
PD_TOP

pt_shell> set_domain_supply_net PD_TOP A \
    -primary_power_net primary_sset.power \
    -primary_ground_net primary_sset.ground \
    -primary_power_net primary_sset.pwell \
    -primary_ground_net primary_sset.nwell

1
pt_shell> set_retention ret_cstr -domain PD_TOP \
    -retention_power_net primary_sset.power \
    -retention_ground_net primary_sset.ground
1
pt_shell> set_isolation iso_cstr -domain PD_TOP \
    -isolation_power_net primary_sset.power \
    -isolation_ground_net primary_sset.ground \
1

```

The following example illustrates how a supply set can be associated with actual supply nets.

```

pt_shell> create_supply_set primary_sset
primary_sset

pt_shell> create_power_domain PD_TOP
PD_TOP

pt_shell> set_domain_supply_net PD_TOP \
    -primary_power_net primary_sset.power \
    -primary_ground_net primary_sset.ground
1
pt_shell> set_retention ret_cstr -domain PD_TOP \
    -retention_power_net primary_sset.power \
    -retention_ground_net primary_sset.ground
1
pt_shell> set_isolation iso_cstr -domain PD_TOP \
    -isolation_power_net primary_sset.power \
    -isolation_ground_net primary_sset.ground \
1
pt_shell> create_supply_net TOP_VDD -domain PD_TOP
TOP_VDD
pt_shell> create_supply_net TOP_VSS -domain PD_TOP
TOP_VSS
pt_shell> create_supply_set primary_sset \

```

```
-function {power TOP_VDD} \
-function {ground TOP_VSS} \
-update
'
1
```

SEE ALSO

```
create_supply_net(2)
set_domain_supply_net(2)
set_retention(2)
set_isolation(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

<i>voltage_area_name</i>	string
<i>coordinate_list</i>	list
<i>polygon_coordinate_list</i>	list
<i>power_domain_name</i>	string
<i>guard_band_width</i>	integer
<i>cell_list</i>	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 user-defined voltage area.

-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.

-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.

-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.

-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 gets the current design in **PrimeTime**.

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 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 found, an error is issued and the working design remains unchanged.

DESCRIPTION

Sets or gets the working design for many PrimeTime commands. Without arguments, the **current_design** command returns a collection containing the current working design. The combination of the current design and current instance defines the context 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"}
```

current_design

168

```
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 in **pt_shell** 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) in **pt_shell**. If the *instance* option is not specified, the focus returns to the top-level of the hierarchy in the current design. If the *instance* option is `.`, the current instance remains unchanged. If the *instance* option is `..`, the context is moved up one level in the instance hierarchy. If the *instance* option begins with `/`, **pt_shell** sets both the current design and the current instance. More complex examples of the *instance* arguments are described in EXAMPLES.

DESCRIPTION

The **current_instance** command sets the working instance in **pt_shell**. 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 context for many **pt_shell** 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 no *instance* argument is specified, the focus of **pt_shell** is returned to the top level of the hierarchy.
- If *instance* is `".`, the current instance is returned and no change is made.
- If *instance* is `..`, the current instance is moved up one level in the design hierarchy.
- If *instance* is a valid cell (or cell name) at the current level of hierarchy, the current instance is moved down to that level of the design hierarchy.

- Multiple levels of hierarchy can be traversed in a single call to **current_instance** by separating multiple cell names with slashes. For example, **current_instance U1/U2** sets the current instance down two levels of hierarchy.
- The "..." directive can also be nested in complex *instance* arguments. For example the command **current_instance ".../MY_INST"** attempts to move the context up two levels of hierarchy, then down one level to the "**MY_INST**" cell.

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'.
```

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_power_rail

Sets or gets the power rails in a multi-rail design to be included in power analysis. As the default (if not specified), all the power rails in the design are included in power analysis. This command has no effect on single rail design.

When the variable `power_enable_multi_rail_analysis` is true, specifies the power rails that will be included in reporting. When false, sets the specific power rails to be included in the power analysis.

SYNTAX

```
int current_power_rail
[rail_names]
```

Data Types

`rail_names` string

ARGUMENTS

`rail_names`

Specifies the names of the power rails in the design to be included in power analysis or reporting. If the `rail_names` option is not specified, the **current_power_rail** command returns a collection of power rails being included in power analysis. If the `rail_names` option refers to a power rail that is not defined in the design, an error is issued and the current rail settings remains unchanged.

DESCRIPTION

Sets or gets the power rails in a multirail design to be included in power analysis or in reporting. Without arguments, the **current_power_rail** command returns a collection of power rails being included in power analysis.

The command behavior depends on the `power_enable_multi_rail_analysis` variable. When false, (the default); concurrent multi-rail analysis is disabled; and power for all the power rails in the design are calculated. This command controls which power rails are to be included in the power analysis. When specified, only the power dissipation on the listed power rails is calculated and reported. To report power consumption per power rail, the command must be applied for each power rail, and the power reanalyzed per power rail.

When the `power_enable_multi_rail_analysis` variable is true, power per power rail is calculated, and the command controls the power reporting and attribute values. When specified, the power dissipation on the listed power rails is accessible via the `get_attribute` and `report_attribute` commands, as well as the `report_power` command. To access the power attribute values per power rail, the command must be applied for each power rail, before issueing the `get_attribute` or `report_attribute` commands.

Before using this command, you define the design power rails by using the **create_power_rail_mapping** command. The **report_power_rail_mapping** command can be used

to show the defined design power rails, library power rails, and rail mapping information.

The following power analysis results will be affected by this command in a multi-rail design:

- average power report
- time-based power report
- power related attributes: total_power
dynamic_power
internal_power
switching_power
leakage_power
x_transition_power
glitch_power
peak_power

Use the **current_power_rail all** command to resume the default behavior, which includes all the power rails in the design.

Peak_power and peak_time per rail are only accessible when power_enable_multi_rail_analysis is false. When concurrent multi-rail analysis is performed, only the peak power and peak time for the all the supply nets is accessible. Per rail the peak power attributes are available only when concurrent multi-rail analysis is disabled.

EXAMPLES

The following example shows generating rail-based power analysis results. The first **report_power** command will generate the power analysis results for the part of design with supply voltage of VDD1 and VDD2. The second **report_power** command will generate the results for the part of design which is supplied by VDD1.

```
pt_shell> create_power_rail_mapping VDD1 -cells block1
pt_shell> create_power_rail_mapping VDD2 -cells block2
pt_shell> current_power_rail {VDD1 VDD2}
pt_shell> report_power
pt_shell> current_power_rail VDD1
pt_shell> report_power
```

SEE ALSO

```
create_power_rail_mapping(2)
report_power_rail_mapping(2)
update_power(2)
report_power(2)
report_power_calculation(2)
```

current_scenario

Selects a subset of the scenarios in the current session for analysis.

SYNTAX

```
int current_scenario
[scenario_list]
[-all]
```

Data Types

scenario_list list

ARGUMENTS

[*scenario_list*]
A list of scenario names in the current session to analyze. If the name of a scenario that exists is specified but that scenario is not in the current session, then the command will flag an error and will not change the command focus. The list can also contain patterns containing wildcard symbols "*" and "?" to select multiple scenarios. When wildcards symbols are used, the scenarios selected are automatically restricted to come from the scenarios in the current session.

[-all]
Selects all scenarios in the current session for analysis.

DESCRIPTION

This command is only available if you invoke the **pt_shell** command with the *-multi_scenario* option.

In multi-scenario analysis, after the scenarios have been created, the **current_session** command is used to focus on a set of scenarios for analysis. The **current_scenario** command is used to focus the analysis on a specific subset of the scenarios in the current session, only those scenarios selected will receive subsequent commands.

The **current_scenario** command returns a collection containing all the scenarios in command focus.

To determine which scenarios are in command focus use the **report_multi_scenario_design** command with the *-session* option.

If the command is executed at remote slaves by **remote_execute** command, it returns the name of the scenario executing the command. In this mode, the options of the command are disabled.

EXAMPLES

In the following example, three scenarios named scen1, scen2 and scen3 are created.

scen1 is created using common_s1.pt and specific_s1.pt scen2 is created using common_s2.pt and specific_s2.pt scen3 is created using common_s3.pt and specific_s3.pt

The scenarios scen1, scen2 and scen3 are selected for the current session which is then displayed. Notice that all three scenarios are in command focus.

The analysis is focused on only scen2 and scen3 using the **current_scenario** command and the session is displayed again. Notice that scen1 is now only in session focus but scen2 and scen3 are in command focus. i.e. only scen2 and scen3 will be considered for analysis.

```
1
pt_shell> create_scenario -name scen1 -common_data {common_s1.pt} \
           -specific_data {specific_s1.pt}
1
pt_shell> create_scenario -name scen2 -common_data {common_s2.pt} \
           -specific_data {specific_s2.pt}
1
pt_shell> create_scenario -name scen3 -common_data {common_s3.pt} \
           -specific_data {specific_s3.pt}
1
pt_shell> current_session {scen1 scen2 scen3}
{"scen1", "scen2", "scen3"}
pt_shell> report_multi_scenario_design -session

*****
Report : multi_scenario_design
Version: V-2004.06
Date   : Tue Apr 13 05:50:10 2004
*****
```

Session details

Number of scenarios in current session: 3

Focus	Scenario (Image provider)
Command	scen1 (scen1)
Command	scen2 (scen2)
Command	scen3 (scen3)

```
1
pt_shell> current_scenario {scen2 scen3}
{"scen2", "scen3"}
pt_shell> report_multi_scenario_design -session

*****
```

Report : multi_scenario_design

current_scenario

```
Version: V-2004.06-Beta2
Date   : Tue Apr 13 05:51:53 2004
*****
```

Session details

```
Number of scenarios in current session: 3
```

Focus	Scenario (Image provider)
-------	---------------------------

Session	scen1 (scen1)
Command	scen2 (scen2)
Command	scen3 (scen3)

```
1
```

In the example below, two scenarios named scen1 and scen2 are in focus. The **remote_execute** command is used to save PrimeTime sessions for scenarios scen1 and scen2 in directories named /u/ptms/sessions/sess_scen1 and /u/ptms/sessions/sess_scen2, respectively.

```
pt_shell> sh mkdir /u/ptms/sessions
pt_shell> remote_execute {save_session /u/ptms/sessions/sess_[current_scenario]}
```

Start of Master/Slave Task Processing

```
Started    : Command execution in scenario 'scen2'
Started    : Command execution in scenario 'scen1'
Succeeded  : Command execution in scenario 'scen2'
Succeeded  : Command execution in scenario 'scen1'
```

End of Master/Slave Task Processing

```
1
pt_shell> ls /u/ptms/sessions
. . . sess_scen1 sess_scen2
pt_shell>
```

SEE ALSO

```
current_session(2)
remove_scenario(2)
report_multi_scenario_design(2)
```

current_session

Selects a set of scenarios for analysis.

SYNTAX

```
string current_session
[scenario_list]
[-all]
```

Data Types

scenario_list list

ARGUMENTS

scenario_list
Brings the specified scenarios into focus for analysis. To specify multiple scenarios, you can use the * and ? wildcard symbols.

-all
Brings all defined scenarios into focus for analysis.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-multi_scenario** option.

To set the current session, you need the following resources:

- At least one PrimeTime license for slave usage; to query the number of PrimeTime licenses available for slave usage, use the **report_multi_scenario_design** command
- At least one host has been added; to query the number of hosts that have been added, use the **report_host_usage** command
- At least one host is online; to query the number of hosts that have come online, use the **report_host_usage** command

After all required resources are available, and the scenarios are created, you must select a set of scenarios to analyze by using the **current_session** command. Calling this command with a list of scenarios brings those scenarios into focus for the current session (session focus) and into focus for receiving commands (command focus) from the master. The current session can be set with all or a subset of the created scenarios.

The **current_session** command returns a collection containing all the scenarios in the current session.

Calling the **current_session** command resets all scenarios to the way they were at the start. After any call to **current_session**, all scenarios in focus are rebuilt from the scripts or images used to originally define them.

EXAMPLES

In the following example, a configuration and three scenarios named scen1, scen2 and scen3 are created. The current session is set such that scen2 and scen3 are in session and command focus.

- scen1 is created using common_s1.pt and specific_s1.pt
- scen2 is created using common_s2.pt and specific_s2.pt
- scen3 is created using common_s3.pt and specific_s3.pt

The scen2 and scen3 scenarios are brought into focus.

```
pt_shell> create_scenario -name scen1 -common_data {common_s1.pt} \
           -specific_data {specific_s1.pt}
1
pt_shell> create_scenario -name scen2 -common_data {common_s2.pt} \
           -specific_data {specific_s2.pt}
1
pt_shell> create_scenario -name scen3 -common_data {common_s3.pt} \
           -specific_data {specific_s3.pt}
1
pt_shell> current_session {scen2 scen3}
{"scen2", "scen3"}
```

SEE ALSO

```
current_scenario(2)
remove_scenario(2)
report_host_usage(2)
report_multi_scenario_design(2)
```

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

<i>name</i>	string
<i>mode_list</i>	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

<i>application_name</i>	string
<i>design_name</i>	string
<i>column_list</i>	list
<i>entries</i>	list_of_lists

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]
[-command_group group_name]
[-hide_body]
[-hidden]
[-dont_abbrev]
[-permanent]
```

Data Types

<i>proc_name</i>	string
<i>info_text</i>	string
<i>arg_defs</i>	list
<i>group_name</i>	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.

-command_group *group_name*

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-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.
 - "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."

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:

```
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}}
    }
}
```

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

<i>data_type</i>	string
<i>obj_class</i>	string
<i>attr_name</i>	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

<i>period_value</i>	float
<i>edge_list</i>	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

```
int disconnect_net
net
object_spec | -all
```

Data Types

<i>net</i>	string
<i>object_spec</i>	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*.

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

<i>val</i>	string
<i>lib_cell_pin</i>	string

ARGUMENTS

-rise	Gets rise drive value.
-fall	Gets fall drive value.
-wire_drive	Not supported in PrimeTime.
-piece <i>val</i>	Not supported in PrimeTime.
<i>lib_cell_pin</i>	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_primate_icc_consistency_settings

Sets all recommended variables with values for consistency with IC Compiler.

SYNTAX

```
string enable_primate_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_primate_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_primate_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

<i>lib_cell</i>	string
<i>fanout</i>	int
<i>wire_load_name</i>	string
<i>lib_name</i>	string
<i>transition</i>	float
<i>clock_list</i>	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 bufla1 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/bufla1
```

SEE ALSO

`report_power(2)`

estimate_eco

Estimates delay changes for the **size_cell** and **insert_buffer** commands.

SYNTAX

```
string estimate_eco
[-lib_cells lib_cell]
[-verbose]
[-max]
[-min]
[-rise]
[-fall]
[-inverter_pair]
[-current_library]
[-libraries lib_spec]
[-significant_digits digits]
[-nosplit]
[-type size_cell | insert_buffer]
[-sort_by sort_type]
object_list
```

Data Types

<i>lib_cell</i>	list
<i>lib_spec</i>	string
<i>digits</i>	integer
<i>sort_type</i>	string
<i>object_list</i>	list

ARGUMENTS

-lib_cells <i>lib_cell_name</i>	Specifies the list of library cells to be used for ECO.
-verbose	Specifies that verbose message is to be shown.
-max	Specifies that maximum delay calculation is to be shown.
-min	Specifies that minimum delay calculation is to be shown.
-rise	Specifies that stage driver output pin is rising.
-fall	Specifies that stage driver output pin is falling.
-inverter_pair	Specifies inverter pair is to be inserted when the ECO type is insert_buffer .

```

-current_library
    Searches only the cell's current library to resolve the lib_cell name.

-libraries lib_spec
    Resolves lib_cell from the libraries specified this option only.

-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
    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.

-nosplit
    Prevents line-splitting and facilitates writing software to extract
    information from the report output.

-type size_cell | insert_buffer
    Specifies that either size_cell or insert_buffer ECO estimation type is to
    be performed. If you do not use this option, the default type is size_cell.

-sort_by sort_type
    Specifies the method of sorting library cells to be reported. The following
    types are allowed: area, stage_delay, arrival, slack, transition,
    max_transition, min_transition, max_capacitance, or min_capacitance. If you
    do not use this option, the default type is area.

object_list
    Specifies the name of object. For size_cell, the object should be a cell. For
    insert_buffer, the object should be a pin.

```

DESCRIPTION

The **estimate_eco** command provides a method to estimate the timing effects of certain ECO commands, without actually incurring a timing update to compute the slack. As its name implies, the results of this command are estimates only, and are expected to reasonably predict, but not necessarily match, the actual slack resulting from the ECO command.

ECO estimation is supported for two ECO commands: **size_cell** and **insert_buffer**. This is specified with the **-type** argument, which takes a value of **size_cell** or **insert_buffer**, which matches the corresponding command name. Both buffer insertion and double-inverter insertion are supported for the **insert_buffer** estimation.

When performing **insert_buffer** estimation, the **-lib_cell** option is required and must supply a list of one or more buffer library cells as buffer insertion candidates. The buffer cells can be specified in the form of lib_cell collection objects, a list of library_name/base_name names, or simply a list of base_name library cell names. The library cell specifications follow the same rules as the **insert_buffer** command. For more information, see the **insert_buffer** man page.

When performing **size_cell** estimation, the **-lib_cell** option is optional and defaults

to the list of alternative library cells for the specified instance cell as obtained by the **get_alternative_lib_cells** command. When the **-lib_cell** option is specified, the candidate library cells can be specified in the form of lib_cell collection objects, a list of library_name/base_name names, or simply a list of base_name library cell names. The library cell specifications follow the same rules as the **size_cell** command. To improve runtime, manually specified candidate cells are not validated for functional equivalence before estimation is performed. For more information, see the **size_cell** man page.

For both commands, if you want to limit the candidate cells to a specific library, you can use the **-current_library** and **-libraries** options of the **get_alternative_lib_cells** command, and pass the resulting lib_cell collection to the **estimate_eco** command.

The **estimate_eco** command generates a report that predicts the timing and slack effects of one or more potential library cell candidates for the **size_cell** or **insert_buffer** command. By default, the estimates are shown in a compact form that lists one candidate library cell per line, with stage delay, arrival and slack information columns. A verbose mode is always available with the **-verbose** option. In verbose mode, a separate detailed estimation report is shown for each candidate library cell.

For example, in the verbose mode, a stage delay is divided into a cell delay, an output net delay, a buffer delay, and a delta delay. The cell delay indicates the cell arc delay of the driver cell of the net. The output net delay is the net arc delay from the driver pin of the net to the slack-critical load pin. The buffer delay is displayed only for buffer insertion estimate and indicates the cell arc delay of the buffer cell. The delta delay indicates the crosstalk delta delay of the net.

The **-min** and **-max** options are available to specify whether minimum-delay hold timing, maximum-delay setup timing, or both should be shown. By default, the maximum-delay timing is shown. The **-rise** and **-fall** options are available to specifically request rise/fall timing. By default, both rise and fall information is shown. If only the **-rise** or **-fall** option is specified, only that information is shown.

The estimate computation takes into account the incoming transition times, the output loading, the fanout loading, the detailed parasitics (if present) and the driver characteristics of the candidate cells. The rise/fall timing printed in the report represents the worst timing through the cell output pins in the corresponding direction. In the verbose report, the arrival is the arrival time at the input pin of the load cell driven by the resized or inserted cell, and the stage delay is the delay of the inserted/resized cell and the driven net.

In the compact report without the verbose option, the reported arrival time, slack and transition time are the estimated values at the input pin of the load cell. Maximum transition, minimum transition, maximum capacitance, and minimum capacitance values are the worst value of the pins in the stage. Individual values can be seen in the verbose reporting.

To perform slack and arrival estimations, the **timing_save_pin_arrival_and_slack** variable should be set to **true** when this command is issued. If the **timing_save_pin_arrival_and_slack** 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, thus preventing the cost of an additional timing update.

Estimated data can be selectively reported if the content of the **eco_estimation_output_columns** variable is changed. By default, only **area**, **stage_delay**, **arrival**, and **slack** are reported. Others, such as **max_transition** and **max_capacitance** can be reported by changing the variable. For more information, see the **eco_estimation_output_columns** man page.

EXAMPLES

The following example shows a minimum and maximum delay summary estimation of four potential buffer cell insertions at an instance pin:

```
pt_shell> estimate_eco -type insert_buffer -min -max \
    -lib_cells {BUFX2 BUFX4 DLY1X1 DLY2X1} \
    wishbone/TxDone_wb_reg/D

delay type : max_rise
lib cell      area  stage_delay   arrival   slack
-----
slow/DLY1X1   19.96    0.857     2.076    14.859
slow/DLY2X1   19.96    1.218     2.437    14.498
slow/BUFX4    16.63    0.639     1.858    15.077
slow/BUFX2    13.31    0.647     1.867    15.069
*No buffer    0.00     0.491     1.711    15.225

delay type : max_fall
lib cell      area  stage_delay   arrival   slack
-----
slow/DLY1X1   19.96    0.746     1.966    14.767
slow/DLY2X1   19.96    1.088     2.308    14.425
slow/BUFX4    16.63    0.514     1.734    14.999
slow/BUFX2    13.31    0.534     1.754    14.979
*No buffer    0.00     0.354     1.573    15.160

delay type : min_rise
lib cell      area  stage_delay   arrival   slack
-----
slow/DLY1X1   19.96    0.366     2.076    0.296
slow/DLY2X1   19.96    0.727     2.437    0.657
slow/BUFX4    16.63    0.148     1.858    0.077
slow/BUFX2    13.31    0.156     1.867    0.086
*No buffer    0.00     0.000     1.711    -0.070

delay type : min_fall
lib cell      area  stage_delay   arrival   slack
-----
slow/DLY1X1   19.96    0.679     1.966    0.098
slow/DLY2X1   19.96    1.021     2.308    0.440
slow/BUFX4    16.63    0.447     1.734    -0.134
slow/BUFX2    13.31    0.467     1.754    -0.114
*No buffer    0.00     0.286     1.573    -0.294
```

The following example shows a minimum delay verbose estimation of a single potential buffer cell insertion at an instance pin:

```
pt_shell> estimate_eco -type insert_buffer \
    -min -verbose -lib_cells {DLY1X1} \
    wishbone/TxDone_wb_reg/D
```

cell name	:	wishbone/TxDoneSync1_reg
current lib cell:		slow/DFFRHQX1
buffer lib cell: slow/DLY1X1		
min_rise	current	estimate

input net delay	0.012	0.012
stage delay	0.000	0.366
cell delay	0.000	0.000
output net delay	0.000	0.000
buffer delay	0.000	0.366
delta delay	0.000	0.000
arrival time	1.711	2.076
slack	-0.070	0.296
buffer lib cell: slow/DLY1X1		
min_fall	current	estimate

input net delay	0.011	0.011
stage delay	0.286	0.679
cell delay	0.286	0.286
output net delay	0.000	0.000
buffer delay	0.000	0.392
delta delay	0.000	0.000
arrival time	1.573	1.966
slack	-0.294	0.098

The following example shows a maximum delay summary estimation of potential cell instance resizes for a given inverter cell name pattern:

```
pt_shell> estimate_eco -type size_cell \
    -lib_cells {INV*} wishbone/bd_ram/U9/U6200
```

delay type : max_rise	lib cell	area	stage_delay	arrival	slack

slow/INVX20	63.20	1.062	9.340	0.019	
slow/INVX16	56.55	1.173	9.451	-0.092	
slow/INVX12	43.24	1.347	9.624	-0.266	
slow/INVX8	19.96	1.682	9.961	-0.602	
*slow/CLKINVX4	13.31	2.727	11.005	-1.646	
slow/INVX3	13.31	3.423	11.701	-2.342	
slow/INVX4	13.31	2.770	11.048	-1.689	
slow/INVX2	9.98	4.943	13.220	-3.861	
slow/INVX1	6.65	9.269	17.546	-8.187	
slow/INVXL	6.65	13.239	21.516	-12.157	

delay type : max_fall	lib cell	area	stage_delay	arrival	slack
<hr/>					
slow/INVX20		63.20	0.855	7.542	1.843
slow/INVX16		56.55	0.940	7.627	1.758
slow/INVX12		43.24	1.074	7.761	1.624
slow/INVX8		19.96	1.279	7.967	1.418
*slow/CLKINVX4		13.31	2.731	9.418	-0.033
slow/INVX3		13.31	2.278	8.965	0.420
slow/INVX4		13.31	1.901	8.588	0.797
slow/INVX2		9.98	3.192	9.879	-0.494
slow/INVX1		6.65	5.420	12.107	-2.722
slow/INVXL		6.65	7.474	14.161	-4.776

The following example shows a maximum delay verbose estimation of potential cell instance resize for specific user-specified alternative cells:

```
pt_shell> estimate_eco -type size_cell -verbose \
           wishbone/bd_ram/U9/U6200 -lib_cells {INVX20}

cell name      : wishbone/bd_ram/U9/U6200
current lib cell: slow/CLKINVX4

new lib cell: slow/INVX20
max_rise          current        estimate
-----
area              13.310       63.200
input net delay   0.005       0.005
stage delay       2.727       1.062
  cell delay      2.609       0.957
  output net delay 0.118       0.105
  delta delay     0.000       0.000
arrival time      11.005      9.340
slack             -1.646      0.019

new lib cell: slow/INVX20
max_fall          current        estimate
-----
area              13.310       63.200
input net delay   0.005       0.005
stage delay       2.731       0.855
  cell delay      2.612       0.751
  output net delay 0.118       0.104
  delta delay     0.000       0.000
arrival time      9.418       7.542
slack             -0.033      1.843
```

The following example shows transition and maximum transition only by changing the variable.

```
pt_shell> set eco_estimation_output_columns "transition max_transition"
pt_shell> estimate_eco -type size_cell -max -rise ram/U9/U6200

delay type : max_rise
lib cell      trans    max_trans
```

```

-----
slow/INVX20          9.340      0.019
slow/INVX16          9.451      -0.092
slow/INVX12          9.624      -0.266
slow/INVX8           9.961      -0.602
*slow/CLKINVX4       11.005     -1.646
slow/INVX3           11.701     -2.342
slow/INVX4           11.048     -1.689
slow/INVX2           13.220     -3.861
slow/INVX1            17.546    -8.187
slow/INVXL           21.516    -12.157

```

The following example shows how the slack value can be parsed from any estimate_eco verbose report for a given min/max rise/fall condition:

```

# specify lib_cell and design instance
set lib_cell INVX20
set instance wishbone/bd_ram/U9/U6200

# obtain all min/max rise/fall estimation info for a single ECO
redirect -variable rpt {estimate_eco -type size_cell -min -max \
-rise -fall -verbose -lib_cells $cell $instance}

# specify min/max rise/fall conditions of interest
set mm max
set rf rise

# obtain min/max rise/fall subreport
regexp "(${mm}}_${rf}.+?)\n\n" $rpt dummy subrpt

# obtain slack from subreport
regexp {slack +[^ ]+ +([^\n]+)} $subrpt dummy slack
echo $slack

```

This report parsing example can be used with both **size_cell** and **insert_buffer** estimation.

SEE ALSO

```

get_alternative_lib_cells(2)
report_constraint(2)
report_delay_calculation(2)
report_timing(2)
eco_estimation_output_columns(3)

```

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
```

```
5
```

SEE ALSO

[quit\(2\)](#)

extract_model

Generates a timing model or a timing and power model from the gate-level netlist of a design.

SYNTAX

```
string extract_model
[-context_borrow]
[-latch_level levels]
-output file_name
[-format format_list]
[-script_format ptsh | dcsh]
[-parasitic_format format_list]
[-library_cell]
[-remove_internal_arcs]
[-ignore_boundary_parasitics]
[-test_design]
[-validate valid_list]
[-block_scope]
[-block_scope_only]
[-scope_scenario scenario_name]
[-arc_types arc_list]
[-noise]
[-power]
```

Data Types

<i>levels</i>	int
<i>file_name</i>	string
<i>format_list</i>	list
<i>scenario_name</i>	string
<i>arc_list</i>	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.

-parasitic_format format_list

Specifies a list of formats for the boundary parasitic file written for the model. The allowed values are **spf** (the default) for the SPEF format and **sbpf** for the SBPF format.

-library_cell

Indicates that the model is to be generated as a library cell, named *current_design_name*, rather than a design containing a core library cell. This option eliminates boundary nets, and the boundary parasitics become part of the model. You must use this option if you use the **-remove_internal_arcs** or **-test_design** option.

-remove_internal_arcs

Indicates that internal arcs are to be removed; only the pin-to-pin timing is to be preserved. Internal clocks are lost because internal arcs are needed to support them. The generated model contains only timing arcs that start or end at an input port or output port. If you use the **-remove_internal_arcs** option, you must also use the **-library_cell** option.

-ignore_boundary_parasitics

Indicates that detailed parasitic annotations present on the boundary nets of the design are not to be included in the model. If specified without the **-library_cell** option, the **-ignore_boundary_parasitics** option disables the writing of the SPEF file. If specified with the the **-library_cell** option, the **-ignore_boundary_parasitics** option causes the effects of the boundary net parasitics to be excluded from the 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 only allowed value is **timing**. 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.

-block_scope

Indicates that the block scope information needs to be captured in addition to extracting the timing model. When specified, PrimeTime creates a separate file that stores the scope information based on the current constraint setup of the block for model extraction.

-block_scope_only

Indicates that only the scope information based on the current setup needs to be captured for the block, no need to actually extract the timing model. This option is useful for the later runs when an ETM was already generated but the scope of block-level validation has changed yet the original ETM is not impacted by the change and no need to be regenerated. When specified, PrimeTime creates a file contains scope information captured.

-scope_scenario scenario_name

Specifies the name of the scenario for which the scope data needs to be labeled and later checked for. The scope information captured and stored in the scope data file is marked as corresponding to the given scenario. If not specified, a fixed default name is used internally.

-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. This option is available only when you specify the **lib** format.

-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. This option is only available when you specify the **lib** format.

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.

By default, the generated model is a design containing a single instance of the core cell. All boundary nets in the original design are preserved in the model. The name given to core lib_cell is *current_design name_core* and the DB model design name is *current_design name*. The design is written in db format, to the *filename.db* file, regardless of the values given with the **-format** option.

When scope of the block is captured, it is not affected by the **hier_scope_check_defaults** environment variable. Instead, all applicable scoping information of the block based on its current constraint setup is captured and stored.

You can also use the command to generated scope information for the block that is replaced with the ETM at top-level.

EXAMPLES

The following example generates a model in Synopsys database format 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.db and DMA_model_lib.db.

```
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. Also, it creates scope information for the block.

```
pt_shell> extract_model -library_cell \
           -test_design -output model2 -format {db} -block_scope
```

SEE ALSO

```
check_block_scope(2)
extract_model_capacitance_limit(3)
extract_model_clock_transition_limit(3)
extract_model_data_transition_limit(3)
extract_model_enable_report_delay_calculation(3)
extract_model_gating_as_nochange(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_status_level(3)
extract_model_use_conservative_current_slew(3)
extract_model_with_3d_arcs(3)
extract_model_with_clock_latency_arcs(3)
hier_modeling_version(3)
hier_scope_check_defaults(3)
timing_clock_gating_propagate_enable(3)
timing_disable_clock_gating_checks(3)
timing_enable_preset_clear_arcs(3)
```

filter

The **filter** command, a synonym for the **filter_collection** command, is a Design Compiler emulation command provided for compatibility between PrimeTime and Design Compiler. You use the **filter_collection** command to filter an existing collection, resulting in a new collection. The base collection remains unchanged.

SEE ALSO

`filter_collection(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

<i>collection1</i>	collection
<i>expression</i>	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

<code>==</code>	Equal
<code>!=</code>	Not equal
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
<code><=</code>	Less than or equal to
<code>=~</code>	Matches pattern
<code>!~</code>	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)"]
```

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

<i>type</i>	string
<i>object_list</i>	list

ARGUMENTS

<i>type</i>	Specifies the object class to find. The type value can be <i>design</i> , <i>port</i> , <i>net</i> , <i>cell</i> , <i>pin</i> , <i>clock</i> , <i>library</i> , <i>lib_cell</i> , or <i>lib_pin</i> .
<i>-hierarchy</i>	Find objects throughout the hierarchy.
<i>-flat</i>	Not supported by PrimeTime.
<i>object_list</i>	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

<i>pattern_string</i>	string
<i>scope</i>	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)

find_objects

228

fix_eco_drc

Fixes or improves design rule checking (DRC) violations through netlist changes.

SYNTAX

```
status fix_eco_drc
-type max_transition | max_capacitance | max_fanout
[-methods method_list]
[-buffer_list allowable_buffers]
[-verbose]
[-current_library]
[-setup_margin margin]
[-hold_margin margin]
[-physical_mode none | open_site | occupied_site]
[object_list]
```

Data Types

<i>method_list</i>	list
<i>allowable_buffers</i>	list
<i>margin</i>	float
<i>object_list</i>	list

ARGUMENTS

```
-type max_transition | max_capacitance | max_fanout
    Specifies the violation type to be fixed.

-methods method_list
    Specifies a fixing method type. Available values are size_cell and
    insert_buffer. You can specify either one of them or both. The default is to
    use both, as in {size_cell insert_buffer}.

-buffer_list allowable_buffers
    Specifies the list of allowable buffers to use for fixing with insert_buffer.
    This option takes only library cell base names, not collection or full library
    cell names. PrimeTime searches for the library cell in the libraries
    contained within the link_path variable. This option is required if you use
    the -methods insert_buffer option. If a library cell is specified by this
    option, PrimeTime uses the library cell for fixing even if it has the dont_use
    attribute; this is to honor the intention of the user who triggered the
    command. If you use the -methods {size_cell insert_buffer} option, the tool
    may size the inserted buffers to buffers that are not specified by the -
    buffer_list option.

-verbose
    Shows additional information during the fixing process. Upon completion, it
    shows detailed information about the remaining violations as well as any
    unfixable reasons for those that are not fixed. This option is recommended
    for interactive ECO mode since it can guide you to the next steps.
```

-current_library
 Specifies the use of library cells from only the current library that the cell belongs to when **size_cell** is specified for the **-methods** option. This can improve runtime but might degrade the quality of results, as the number of available alternative library cells for sizing is reduced.

-setup_margin margin
 Specifies the margin applied to setup slacks during DRC fixing. The **fix_eco_drc** command tries (but cannot guarantee) to preserve the setup slack by the specified amount while it fixes DRC violations. By default, the setup margin is negative infinity; that is, the **fix_eco_drc** command fixes violations even if setup slack is degraded. For example, a margin of zero specifies that the **fix_eco_drc** command does not fix a DRC violation if the fix would degrade the setup slack below zero. Specify the margin value in library units.

-hold_margin margin
 Specifies the margin applied to hold slacks during DRC fixing. The **fix_eco_drc** command tries (but cannot guarantee) to preserve the hold slack by the specified amount while it fixes DRC violations. By default, the hold margin is negative infinity; that is, the **fix_eco_drc** command fixes violations even if hold slack is degraded. For example, a margin of zero specifies that the **fix_eco_drc** command does not fix a DRC violation if the fix would degrade the hold slack below zero. Specify the margin value in library units.

-physical_mode none | open_site | occupied_site
 Specifies the use of physical data to resize cells or place buffers with one of the following modes:

- **none** (the default) - Does not use physical data while fixing violations.
- **open_site** - Sizes a cell only if there is enough room available around the cell and inserts a buffer only if there is a free empty site is available. This mode intends to not move existing cells and place cells only in available free sites.
- **occupied_site** - Considers layout density in the cell neighborhood and places a cell overlapping existing neighbor cells if the density is low enough. This mode intends to place a cell even if no free site is available and depends on a place-and-route tool to move existing cells to create a room for the placed cell.

object_list
 Specifies a list of pin or port objects in the current design for DRC fixing. If you do not specify the **object_list**, the tools tries to fix all DRC violations in the current design.

DESCRIPTION

This command seeks to make ECO changes to the design to improve or resolve any DRC violations. The supported violation types are **max_transition**, **max_capacitance** and **max_fanout**. You can use this command in the single-core analysis, distributed multi-scenario analysis (DMSA), and multicore analysis flows within PrimeTime.

You must specify a **max_transition**, **max_capacitance** or **max_fanout** violation type with the **-type** option. To fix violations, driver cell resizing or buffer insertion is used to improve transition time for **max_transition** violations, driving strength for **max_capacitance** and **max_fanout** violations.

If some violations cannot be fixed and the **-verbose** option is specified, this command shows why they cannot be fixed. See the following detailed unfixable reasons. An example of an unfixable reason is that no alternative cells exist with enough drive capability to resolve the violation.

During the fixing process, the **dont_use** attribute of library cell is honored. However, when you specify library cells with the **-buffer_list** option, PrimeTime uses all the specified library cells for fixing regardless of their **dont_use** attribute. Use **set_dont_use** command to apply the **dont_use** attribute to certain library cells so that they are not used during fixing. In addition, use of library cells can be also restricted by applying the **pt_dont_use** user-defined attribute on the restricted lib_cell objects, which is the legacy feature. For an example of how to apply this attribute, see the EXAMPLES section of the **fix_eco_timing** command.

Note that the **fix_eco_drc** command is designed for the sign-off flow with parasitics, in which delays and slews are calculated based on parasitics. If delays and slews are not computed inside PrimeTime but annotated such as in Standard Delay Format (SDF) flow, the quality of results cannot be guaranteed.

Selecting the Fixing Method

If you specify only **size_cell** as your fixing method, this command resizes the driver cell of the stage with a violation. If a violation is at a cell input pin, a driver cell to this pin is resized. If a violation is at a cell output pin, its cell, which is also a driver of the stage, is resized. The **eco_alternative_area_ratio_threshold** variable determines alternative library cells to be considered for resizing. If this variable is nonzero, increasing the value of this variable results in more library cells allowed for resizing. If this variable is zero, all alternative library cells are considered. A sequential cell is not resized because it can potentially unbalance clock trees and create unexpected new violations.

If you specify only the **insert_buffer** argument as your fixing method, this command inserts a buffer to remove the violation.

If you specify both the **size_cell** and **insert_buffer** arguments, this command chooses the best fixing method to remove the violation and minimizes the area impact. With this method, **size_cell** does not honor the **eco_alternative_area_ratio_threshold** variable because more layout perturbation is permissible.

***dont_touch* and UPF**

Cells with the **dont_touch** attribute, specified directly or implicitly on hierarchical cells, are not considered for resizing to fix violations. Similarly, nets with the **dont_touch** attribute, specified directly or implicitly, are not considered for buffer insertion.

If a cell has the **always_on**, **is_isolation**, **is_retention**, or **is_level_shifter** low-power control attribute set to **true**, it is not considered for resizing. Buffer

insertion is not allowed for the net connected to all pins of an always_on cell, control and clock pins of a retention cell. Buffer insertion is also prohibited for the nets between a port and an isolation cell or between a port and a level shifter cell.

Timing Margin

By default, the **fix_eco_drc** command attempts to fix a DRC violation even if it creates a new setup or hold timing violation.

If you want to fix DRC violations but keep setup or hold violation at certain value, use the **-setup_margin** and **-hold_margin** options to allow timing margins. If you specify a negative setup or hold margin, you are relaxing timing slack requirements and allowing the command to fix DRC violations at the expense of timing slack. For example, if you specify -1.0 for the setup margin, this command can fix DRC violations, but might create new setup violations with slacks close to -1.0.

On the contrary, if you specify a positive setup or hold margin, this command fixes violations with tighter timing margins. This results in less freedom to fix DRC violations and might result in unresolved DRC violations.

Unfixable Violations

If you specify **fix_eco_drc** with the **-verbose** option, the tool reports the unfixed violations with the following explanations. For more information, run the **estimate_eco** command on specific cells to see the details of estimated DRC violations, delays, and slacks.

- A - There are available lib cells outside area limit**

This reason appears if you set **eco_alternative_area_ratio_threshold** variable to limit the area of alternative library cells, and there are other alternative library cells with larger area, which could be considered for fixing. You might be able to fix the violation by increasing the area limit.

- C - The violation is in clock network**

This reason appears if the violation is in a clock network. Any violation in a clock network is not fixed because it can change the clock tree timing and potentially create new timing violations.

- D - Cell or net is located in high density area**

This reason appears if there is not enough room available in the vicinity of the stage. Inspect your LEF/DEF file for filler cells and validate whether they can be and are treated as **open_sites**. When the **eco_allow_filler_cells_as_open_sites** variable is set to true, physically-aware ECO commands treat all filler cells as **open_sites**. This variable is by DEFAULT set to true. Additionally, place-and-route tool can also be used to reduce the region density.

- E - Physical information is incomplete or unavailable**

This reason appears if physical data is not available or incomplete for the specific

net or cell. Check if the DEF and LEF files are consistent with your current design.

- **H - Logical and physical hierarchy is inconsistent**

This reason appears if the netlist and physical layout are not consistent, and a buffer cannot be inserted. Use place-and-route tool to resolve inconsistency.

- **I - Buffer insertion with given lib cells cannot fix the violation**

This reason appears if the violation cannot be fixed with given buffer library cells specified by the **-buffer_list** option. Add more buffer library cells, if needed.

- **O - No open free site is available**

This reason indicates that there is no empty free site available on the route or in the vicinity of the specified pin or cell. Inspect your LEF/DEF file for filler cells and validate whether they can be and are treated as `open_sites`. When the `eco_allow_filler_cells_as_open_sites` variable is set to true, physically-aware ECO commands treat all filler cells as `open_sites`. This variable is by DEFAULT set to true. Additionally, place-and-route tool can also be used to create free sites.

- **P - Driver cell of the violation is a port**

This reason indicates that the driver of the stage in which violation exists is a port, thus, resizing cannot be applied. Use a buffer insertion method to fix the violation.

- **Q - Driver cell of the violation is a sequential cell**

This reason indicates that the driver of the stage in which violation exists is a sequential cell, and resizing cannot be applied. The **fix_eco_drc** command does not resize sequential cells because it can potentially change clock network timing and create new timing violations.

- **S - Cell sizing with alternative lib cells cannot fix the violation**

This reason indicates that all alternative library cells have been considered for resizing to fix the violation, but the violation cannot be fixed. Use a buffer insertion method to fix the violation, if needed.

- **T - Timing margin is too tight to fix the violation**

This reason indicates that the violation is on a critical or near-critical setup or hold timing path and does not have enough timing slack to fix the violation. You can relax this by providing more timing margins and specifying negative values for the **-setup_margin** and **-hold_margin** options.

- **V - Driver cell of the violation is dont_touched**

This reason appears if the driver cell is `dont_touched` or low power control attributes, such as `always_on`, `is_retention`, `is_isolation`, and `is_level_shifter`.

This attribute also appears if the connected net has the `dont_touch` attribute set to `true`. See the UPF section of this man page for more details.

When the command finishes, the following remaining violations are displayed:

Remaining Violations:

Violation Type	Count
<hr/>	
Total remaining violations	10
Unfixable violations	8

This table shows that there are eight unfixable violations. The details are shown if you specify the **-verbose** option. There are two violations that can be fixed if the **fix_eco_drc** command is executed again. It indicates that the violations might possibly be fixable with an additional run, but it does not guarantee that all of them are fixed.

Moreover, you might want to try other options, such as relaxing timing constraints or increasing the alternative cell area limit, to maximize your QoR.

Fixing Multiple Scenarios With Fewer Hosts

PrimeTime SI requires the same number of host as the number of scenarios because the best fixing rate and fastest runtime can be achieved when the same number of hosts are used. It maximizes parallelism with distributed multi-scenario processing. However, in some cases, you might want to perform ECO fixing with fewer hosts. The **fix_eco_drc** command allows this configuration if the **eco_enable_more_scenarios_than_hosts** variable is set to **true**. Note that this setting can affect performance and QoR as the number of hosts is reduced, and the **fix_eco_drc** command runs in a special mode to optimize computing resource, runtime, and QoR.

If fewer hosts are available than scenarios, the hosts run in scenario-swapping mode, where they rotate and swap scenarios one after the other. This can result in significant performance degradation.

To achieve faster runtimes, you need to minimize the number of merged reporting commands, such as **report_timing** and **report_analysis_coverage**, because they trigger a full round of scenario swapping. Moreover, reduce the number of **remote_exec{}** commands and merge them into one **remote_exec{}** block. For detailed examples, see the **fix_eco_timing** man page.

Physically-Aware DRC Fixing

When you specify **open_site** or **occupied_site** for the **-physical_mode** option, PrimeTime reads the physical data and uses it to fix DRC violations.

- To minimize layout disturbances for your ECO changes, such as in the late stage of your ECO cycle, use the **open_site** mode. This prevents PrimeTime from moving existing cells in the layout.
- To maximize your fix rate, such as in the early stage of your ECO cycle, use the **occupied_site** mode. This allows PrimeTime to resize existing cells and place new cells when the local placement density can accommodate the change.

For more details, see the man page for the **fix_eco_timing** command.

Fixing with Multiply Instantiated Modules (MIM)

When the **eco_enable_mim** variable is true, PrimeTime automatically derives the MIM configuration. In physically-aware fixing, the MIM configuration is derived from DEF files. If multiple instances share the same DEF file, PrimeTime recognizes them as MIM instances. In non physically-aware fixing, MIM configuration is derived from parasitics files. If multiple instances share the same parasitics file using the **-path** option in the **read_parasitics** command, PrimeTime recognizes them as MIM instances.

For more details, see the man page for the **fix_eco_timing** command.

EXAMPLES

In the following example, you first try to fix maximum transition violations with only the **size_cell** method, and you find that there are two unfixable violations remaining.

```
pt_shell> fix_eco_drc -type max_transition -verbose \
    -methods size_cell
```

Unfixable violations:

- A - There are available lib cells outside area limit
- C - The violation is in clock network
- I - Buffer insertion with given lib cells cannot fix the violation
- P - Driver cell of the violation is a port
- Q - Driver cell of the violation is a sequential cell
- S - Cell sizing with alternative lib cells cannot fix the violation
- T - Timing margin is too tight to fix the violation
- V - Driver cell of the violation is dont_touched

Violation	Reasons
U1/I	P
U6/I	S

Remaining Violations:

Violation Type	Count
Total remaining violations	2
Unfixable violations	2

The unfixable reason "S" shows that U6/I was not fixed because any of the alternative library cells cannot fix the violation within timing slack. The reason "P" shows that U1/I was not fixed because its driver is a port, and resizing cannot be applied.

In the next run, you try the buffer insertion method to fix the violation with the three buffer library cells, BUFX1, BUFX2 and BUFX3.

```
pt_shell> fix_eco_drc -type max_transition -verbose \
    -methods insert_buffer -buffer_list { BUFX1 BUFX2 BUFX3 }
```

Unfixable violations:

A - There are available lib cells outside area limit
 C - The violation is in clock network
 D - Cell or net is located in high density area
 E - Physical information is incomplete or unavailable
 I - Buffer insertion with given lib cells cannot fix the violation
 O - No open free site is available
 P - Driver cell of the violation is a port
 Q - Driver cell of the violation is a sequential cell
 S - Cell sizing with alternative lib cells cannot fix the violation
 T - Timing margin is too tight to fix the violation
 V - Driver cell of the violation is dont_touched

Violation	Reasons
U6/I	I

Remaining Violations:

Violation Type	Count
Total remaining violations	1
Unfixable violations	1

However, this could not fix all the violations. Even though it removed the first violation, it could not fix the second one because the provided buffer library cells are not strong enough to fix the last remaining violation. In the next run, you allow more library buffer cells. Furthermore, you specify both **size_cell** and **insert_buffer** methods to maximize your fixing chance.

The following example shows the unfixed reason report for physical-aware fixing. There are four unfixed reason specific to physical-aware fixing, which is not shown in logical unfixed reason report. The unfixable reason "O" shows that U1/I was not fixed because there is no empty free site available.

```
pt_shell> fix_eco_drc -type max_transition -verbose\
           -methods insert_buffer -buffer_list { BUFX1 BUFX2 BUFX3 } \
           -physical_mdoe open_site
```

Unfixable violations:

A - There are available lib cells outside area limit
 C - The violation is in clock network
 D - Cell or net is located in high density area
 E - Physical information is incomplete or unavailable
 H - Logical and physical hierarchies are inconsistent
 I - Buffer insertion with given lib cells cannot fix the violation
 O - No open free site is available
 P - Driver cell of the violation is a port
 Q - Driver cell of the violation is a sequential cell
 S - Cell sizing with alternative lib cells cannot fix the violation
 T - Timing margin is too tight to fix the violation
 V - Driver cell of the violation is dont_touched

Violation	Reasons
U1/I	O

Remaining Violations:

Violation Type	Count
<hr/>	
Total remaining violations	1
Unfixable violations	1

The following example shows that you have remaining violations that can be fixed in the next runs. You might want to consider running the **fix_eco_drc** command multiple times to remove them.

Remaining Violations:

Violation Type	Count
<hr/>	
Total remaining violations	2
Unfixable violations	0

The following example shows the output example of remaining violations when the **fix_eco_drc** command is running in distributed multi-scenario analysis. In this example, there are two scenarios, scen1 and scen2. There is one unfixable violation in scen1, but the violation in scen2 can be fixed in the next run.

Remaining Violations:

Scenario	Total	Unfixable
<hr/>		
scen1	1	1
scen2	1	0

SEE ALSO

`estimate_eco(2)`
`fix_eco_timing(2)`
`report_analysis_coverage(2)`
`report_constraint(2)`
`report_timing(2)`
`set_dont_use(2)`
`set_eco_options(2)`
`write_changes (2)`
`eco_alternative_area_ratio_threshold(3)`
`eco_enable_more_scenarios_than_hosts(3)`

fix_eco_leakage

Performs leakage recovery driven by signoff timing by replacing lower priority cells with higher priority cells.

SYNTAX

```
string fix_eco_leakage
-priority pattern_list
[-cell_type cell_types]
[-attribute attribute_name]
[-verbose]
[-setup_margin margin]
```

Data Types

<i>pattern_list</i>	list
<i>cell_types</i>	list
<i>attribute_name</i>	string
<i>margin</i>	float

ARGUMENTS

```
-priority pattern_list
    Specifies a list of library cell patterns in the order of highest to lowest
    priority.

-cell_type cell_types
    Specifies a cell type for leakage recovery. Allowed values are combinational
    and sequential. The default is {sequential combinational}, which specifies
    that both sequential and combinational cells are swapped for leakage
    recovery.

-attribute attribute_name
    Specifies an attribute name that can be used to match patterns specified by
    the -priority option. When you specify this option, PrimeTime uses
    the value of the attribute, instead of library cell names, for pattern
    matching.

-verbose
    Shows additional information during leakage recovery process. During
    distributed multi-scenario analysis, the report includes setup violation
    information from each scenario.

-setup_margin margin
    Specifies an additional timing margin to be applied to setup slacks during
    leakage recovery. By default, the setup margin is zero. If you specify a
    positive value, PrimeTime preserves the specified amount of slack during
    leakage recovery. If you specify a negative value, PrimeTime uses the
    specified amount of slack to swap more cells during leakage recovery.
```

DESCRIPTION

This command seeks to recover leakage (reduce leakage current) as much as possible by swapping cells that have positive setup slacks, which can be reduced, without creating new timing violations. You can use the **fix_eco_leakage** command in the multicore analysis and distributed multi-scenario analysis flows within PrimeTime.

Run the **fix_eco_leakage** command in distributed multi-scenario analysis flow to consider signoff timings from all scenarios. PrimeTime analyzes timing across all scenarios during the leakage recovery process and maintains signoff-state timing in all scenarios.

You must specify the required **-pattern_priority** option with a *pattern_list*. This option determines the priority order that PrimeTime follows when replacing lower priority cells with higher priority cells. Therefore, the first pattern indicates the cells with the lowest leakage (highest priority for swapping), and the last pattern indicates the cells with the highest leakage (lowest priority for swapping).

PrimeTime swaps cells if the following conditions are met:

- The original library cell has enough positive setup slack.
- The new library cell is compatible with the original library cell and satisfies the pattern matching criteria (see "Pattern Matching and Priority Order").
- The new library cell has higher priority than the original library cell.
- The new library cell does not create new setup, max_transition, or max_capacitance violations.

Pattern Matching and Priority Order

To determine the priority of cell swapping, the **fix_eco_leakage** command can perform pattern matching if the library cell names have a changing pattern and a fixed pattern. A changing pattern is a text string that represents different leakage values, such as HVT, NVT, and LVT. A fixed pattern is a text string that does not change with different leakage values, such as BUF1X.

To apply the pattern matching algorithm and identify the candidates for cell swapping, use **-pattern_priority** option; specify the cells in the *pattern_list* in the order of highest to lowest priority (most to least preferred).

For example, suppose you have three library cells: HVT_BUF1X, NVT_BUF1X, and LVT_BUF1X. They are the same buffers except with different leakage values. The HVT_BUF1X library cell has the lowest leakage, which is most preferred; the LVT_BUF1X library cell has the highest leakage, which is least preferred.

In this example, you would specify the **-pattern_priority** option with the following *pattern_list*: {HVT NVT LVT}. PrimeTime swaps cells in the following order:

- LVT_BUF1X can be replaced by HVT_BUF1X or NVT_BUF1X.
- NVT_BUF1X can be replaced by HVT_BUF1X.
- HVT_BUF1X cannot be replaced; it has the highest priority.

In this case, HVT_BUF1X has highest priority as the most preferred cell. When the tool finds an LVT_BUF1X cell, it tries to replace it with HVT_BUF1X, if doing so would not create any new timing violations. If not, then the tool tries to replace LVT_BUF1X with NVT_BUF1X. The tools also tries to replace any NVT_BUF1X cell with HVT_BUF1X. However, the tool does not change HVT_BUF1X because it is the most preferred cell.

Specifying the Pattern List

Pattern matching supports prefix, mid-fix, and postfix. For example, you can specify {HVT NVT LVT} for the following library cell names.

```
Prefix : HVT_BUF1X,  LVT_BUF1X,  NVT_BUF1X
Mid-fix: BUF_HVT_1X,  BUF_NVT_1X,  BUF_LVT_1X
Postfix: BUF1X_HVT,  BUF1X_NVT ,  BUF1X_LVT
```

If multiple patterns overlap each other, the largest pattern is chosen. For example, if you specify {LVT LLVT} for the pattern, PrimeTime replaces BUF1X_LLVT (with matching LLVT pattern) by BUF1X_LVT (with matching LVT pattern), even though the LVT pattern also matches BUF1X_LLVT.

Swapping Cells By Attributes Instead of Cell Names

Even if your libraries do not have the naming convention described previously, you can still recover leakage by using a user-defined attribute and the **fix_eco_leakage -attribute** command. This also gives you great flexibility to customize your own flow and apply different techniques in different stages of your design.

For example, suppose that there are six library cells: BUF_A, B_BUF, BF1X, INV_A, B_INV, and IV2X. Although the pattern matching cannot be used with these cell names, you can still recover leakage by using user-defined attributes. In this example, BUF_A and INV_A are the most preferred cells, and BF1X and IV2X are the least preferred cells. To perform cell swapping with attributes, use the following commands:

```
# Define a user-defined attribute 'eco_pattern" for lib cell.
# Assign different value to each library cell with "good ok bad"
#
define_user_attribute eco_pattern -type string -class lib_cell
set_user_attribute -class lib_cell [get_lib_cell BUF_A] eco_pattern "buf_good"
set_user_attribute -class lib_cell [get_lib_cell B_BUF] eco_pattern "buf_ok"
set_user_attribute -class lib_cell [get_lib_cell BF1X ] eco_pattern "buf_bad"
set_user_attribute -class lib_cell [get_lib_cell INV_A] eco_pattern "inv_good"
set_user_attribute -class lib_cell [get_lib_cell B_INV] eco_pattern "inv_ok"
```

```

set_user_attribute -class lib_cell [get_lib_cell IV2X ] eco_pattern "inv_bad"
#
# Specify pattern priority using an attribute
#
fix_eco_leakage -pattern_priority {good ok bad} -attribute eco_pattern

```

Note that the name of user-defined attribute for each library cell must be unique. The name of user-defined attribute for a group of library cells that can be swapped to one another must have a common base name (for example, buf or inv) along with a name to specify priority (for example, good, ok, or bad). This is required for PrimeTime to know which cells can be swapped relative to one another.

Cells That Are Not Swapped

The recovery process honors any **dont_touch** and **dont_use** attributes that are applied to the design. Just as with Design Compiler and IC Compiler, a **dont_touch** attribute can be applied to an upper-level hierarchical cell, and its implicit effect propagates downward into the hierarchy. If a different **true** or **false** **dont_touch** value is applied deeper in the hierarchy, it takes precedence over any higher-level **dont_touch** attributes. Use **set_dont_use** command to apply the **dont_use** attribute to certain library cells so that they are not used during leakage recovery. The recovery process also honors the **pt_dont_use** user-defined attribute as the **fix_eco_timing** command.

The recovery process does not swap any cells in the clock network.

In distributed multi-scenario analysis fixing, a restriction in one scenario also should be applied to other scenarios. A **dont_touch** attribute should be applied to all scenarios to prevent design changes across all scenarios for the affected cells and nets.

Unconstrained Cells

In distributed multi-scenario analysis, a cell is unconstrained if the cell is not constrained in all scenarios. If a cell is not constrained in one scenario but constrained in another scenario, the cell is constrained in distributed multi-scenario analysis.

You can control swapping unconstrained cells by specifying the **eco_leakage_exclude_unconstrained_cells** variable. If the variable is set to **false**, PrimeTime swaps unconstrained cells. If the variable is set to **true**, PrimeTime does not swap unconstrained cells.

UPF

If a cell has the **always_on**, **is_isolation**, **is_retention**, or **is_level_shifter** low-power control attribute set to **true**, it is not considered for cell swapping.

Honoring Footprint Attributes in the Library

If you want PrimeTime to honor a footprint attribute from libraries, you need to import the attribute and specify the attribute name in the

eco_alternative_cell_attribute_restrictions variable. Suppose your library has the **cell_footprint** attribute to indicate cell footprint. Then, the following commands restrict the **fix_eco_leakage** command to swap only footprint-compatible cells.

```
# Import 'cell_footprint' attribute to PrimeTime
define_user_attribute cell_footprint -class lib_cell -import -type string

# Restrict compatible cells only to the same cell_footprint
set eco_alternative_cell_attribute_restrictions {cell_footprint}

# Swappable cells should satisfy both the pattern priority and
# the cell_footprint attribute value
fix_eco_leakage -pattern_priority {HVT NVT LVT}
```

EXAMPLES

The following example shows how to recover leakage with the library cells having prefix HVT, NVT, and LVT for their names. Both sequential and combinational cells are swapped.

```
pt_shell> fix_eco_leakage -pattern_priority {HVT NVT LVT}
```

The following example shows how to swap only combinational cells.

```
pt_shell> fix_eco_leakage -cell_type combinational -pattern_priority {HVT NVT LVT}
```

The following example shows a typical example for the leakage recovery in distributed multi-scenario analysis.

```
#-----
# Update timing & global reporting in master
#-----
remote_execute {
    set timing_save_pin_arrival_and_slack true; update_timing -full
}
set pattern "HVT NVT LVT"
report_global_timing
report_constraint -all -max_cap -max_transition

#-----
# Before leakage recovery:
#   Report timing, power, cell usages for each slave
#-----
remote_execute {
    set pattern "HVT NVT LVT"
    report_global_timing
    report_constraint -all -max_cap -max_trans
    set power_enable_analysis true
    report_power -group {...} -threshold -pattern $pattern
}

#-----
# Before leakage recovery:
#   Check cell swap compatibility. Detect any misuse of patterns
```

fix_eco_leakage

242

```

#     by using the -alternative_lib_cells and -show_others
#     options.
#-----
remote_exec {
    report_cell_usage -pattern $pattern -alt -show_others
}

#-----
# Execute signoff-driven leakage recovery.
#-----
fix_eco_leakage -pattern $pattern -verbose

#-----
# Reporting after leakage recovery
#-----
report_global_timing
report_constraint -all -max_cap -max_transition
remote_execute {
    report_global_timing
    report_constraint -all -max_cap -max_trans
    set power_enable_analysis true
    report_power -group {...} -threshold -pattern $pattern
}
exit

```

SEE ALSO

```

fix_eco_drc(2)
fix_eco_timing(2)
report_cell_usage(2)
set_dont_use(2)
eco_alternative_area_ratio_threshold(3)
timing_save_pin_arrival_and_slack(3)

```

fix_eco_power

Performs area/power recovery driven by signoff timing by downsizing or swapping cells with positive setup slack.

SYNTAX

```
string fix_eco_power
[-cell_type cell_types]
[-setup_margin margin]
[-pattern_priority pattern_list]
[-attribute attribute_name]
[-power_attribute power_attribute_name]
[-verbose]
```

Data Types

<i>cell_types</i>	list
<i>margin</i>	float
<i>pattern_list</i>	list
<i>attribute_name</i>	string
<i>power_attribute_name</i>	string

ARGUMENTS

-cell_type *cell_types*
Specifies a cell type for area/power recovery. Allowed values are **combinational** and **sequential**. The default is **{sequential combinational}**, which specifies that both sequential and combinational cells are considered for area/power recovery.

-setup_margin *margin*
Specifies an additional timing margin to be applied to setup slacks during area/power recovery. By default, the setup margin is zero. If you specify a positive value, PrimeTime preserves the specified amount of slack during area/power recovery. If you specify a negative value, PrimeTime uses the specified amount of slack to downsize or swap more cells.

-pattern_priority *pattern_list*
Specifies a list of library cell patterns in the order of highest to lowest priority.

-attribute *attribute_name*
Specifies an attribute name that can be used to match patterns specified by the **-pattern_priority** option. When you specify this option, PrimeTime uses the value of the attribute, instead of library cell names, for pattern matching.

-power_attribute *power_attribute_name*
Specifies an attribute for prioritizing library cells for downsizing. When you specify this option, PrimeTime uses the value of the power attribute, instead of library cell's area attribute, to prioritize library cells during downsizing, where a library cell with a smaller power attribute value has

higher priority.

-verbose

Shows additional information during area/power recovery process. During distributed multi-scenario analysis, the report includes setup violation information from each scenario.

DESCRIPTION

This command seeks to recover area/power as much as possible by either downsizing or swapping to a cell with higher priority on cells that have positive setup slacks without creating new timing violations. You can use the **fix_eco_power** command in the multicore analysis and distributed multi-scenario analysis flows within PrimeTime.

Run the **fix_eco_power** command in distributed multi-scenario analysis flow to consider signoff timings from all scenarios. PrimeTime analyzes timing across all scenarios during the recovery process and maintains signoff-state timing in all scenarios.

To reduce both area and power by cell downsizing, do not specify the **-pattern_priority** option; the compatible library cells and their priority order are automatically determined by the tool. If you want PrimeTime to reduce leakage power by cell swapping, you need to specify the **-pattern_priority** option with a *pattern_list*. This option determines the priority order that PrimeTime follows when replacing lower priority cells with higher priority cells. Therefore, the first pattern indicates the cells with the lowest leakage (highest priority for swapping), and the last pattern indicates the cells with the highest leakage (lowest priority for swapping).

PrimeTime downsizes or swaps a cell if following conditions are met:

- The cell has enough positive setup slack.
- The new library cell is compatible with the original library cell and satisfies the pattern matching criteria when the **-pattern_priority** option is specified (see "Swapping Cells Using Pattern Matching and Priority Order").
- The new library cell has higher priority than the original library cell.
- The new library cell does not create new setup, max_transition, or max_capacitance violations.

Down-Sizing Cells Using Library Cell Area Attribute

If you do not specify the **-pattern_priority** option, PrimeTime automatically determines candidate library cells from the current library for downsizing. To determine the priority order, PrimeTime uses the value of area attribute of each library cell and prioritize the ones with smaller value. For library cells with the same area, PrimeTime prioritizes a library cell with less impact on timing.

Swapping Cells Using Pattern Matching and Priority Order

PrimeTime performs cell swapping based on pattern matching when you specify the **-pattern_priority** option. Priority of cell swapping is determined by the order of names in the *pattern_list*; you must specify the names in the *pattern_list* in the order of highest to lowest priority (most to least preferred).

In order for pattern matching to work, each library cell name must have a changing pattern and a fixed pattern. A changing pattern is a text string that represents different leakage values, such as HVT, NVT, and LVT. A fixed pattern is a text string that does not change with different leakage values, such as BUF1X.

For example, suppose you have three library cells: HVT_BUF1X, NVT_BUF1X, and LVT_BUF1X. They are the same buffers except with different leakage values. The HVT_BUF1X library cell has the lowest leakage, which is most preferred; the LVT_BUF1X library cell has the highest leakage, which is least preferred.

In this example, you would specify the **-pattern_priority** option with the following *pattern_list*: {HVT NVT LVT}. PrimeTime swaps cells in the following order:

- LVT_BUF1X can be replaced by HVT_BUF1X or NVT_BUF1X.
- NVT_BUF1X can be replaced by HVT_BUF1X.
- HVT_BUF1X cannot be replaced; it has the highest priority.

In this case, HVT_BUF1X has highest priority as the most preferred cell. When the tool finds an LVT_BUF1X cell, it tries to replace it with HVT_BUF1X, if doing so would not create any new timing violations. If not, then the tool tries to replace LVT_BUF1X with NVT_BUF1X. The tools also tries to replace any NVT_BUF1X cell with HVT_BUF1X. However, the tool does not change HVT_BUF1X because it is the most preferred cell.

Specifying the Pattern List

Pattern matching supports prefix, mid-fix, and postfix. For example, you can specify {HVT NVT LVT} for the following library cell names.

```
Prefix : HVT_BUF1X,  LVT_BUF1X,  NVT_BUF1X  
Mid-fix: BUF_HVT_1X,  BUF_NVT_1X,  BUF_LVT_1X  
Postfix: BUF1X_HVT,   BUF1X_NVT ,  BUF1X_LVT
```

If multiple patterns overlap each other, the largest pattern is chosen. For example, if you specify {LVT LLVT} for the pattern, PrimeTime replaces BUF1X_LLVT (with matching LLVT pattern) by BUF1X_LVT (with matching LVT pattern), even though the LVT pattern also matches BUF1X_LLVT.

Swapping Cells By Attributes Instead of Cell Names

Even if your library cells do not have the naming convention described previously,

you can still perform cell swapping by using a user-defined attribute and the **fix_eco_power -attribute** command. This also gives you great flexibility to customize your own flow and apply different techniques in different stages of your design.

For example, suppose that there are six library cells: BUF_A, B_BUF, BF1X, INV_A, B_INV, and IV2X. Although the pattern matching cannot be used with these cell names, you can still perform cell swapping by using user-defined attributes. In this example, BUF_A and INV_A are the most preferred cells, and BF1X and IV2X are the least preferred cells. To perform cell swapping with attributes, use the following commands:

```
# Define a user-defined attribute 'eco_pattern' for lib cell.  
# Assign different value to each library cell with "good ok bad"  
define_user_attribute eco_pattern -type string -class lib_cell  
set_user_attribute -class lib_cell [get_lib_cell BUF_A] eco_pattern "buf_good"  
set_user_attribute -class lib_cell [get_lib_cell B_BUF] eco_pattern "buf_ok"  
set_user_attribute -class lib_cell [get_lib_cell BF1X ] eco_pattern "buf_bad"  
set_user_attribute -class lib_cell [get_lib_cell INV_A] eco_pattern "inv_good"  
set_user_attribute -class lib_cell [get_lib_cell B_INV] eco_pattern "inv_ok"  
set_user_attribute -class lib_cell [get_lib_cell IV2X ] eco_pattern "inv_bad"  
  
# Specify pattern priority using an attribute  
fix_eco_power -pattern_priority {good ok bad} -attribute eco_pattern
```

Note that the name of user-defined attribute for each library cell must be unique. The name of user-defined attribute for a group of library cells that can be swapped to one another must have a common base name (for example, buf or inv) along with a name to specify priority (for example, good, ok, or bad). This is required for PrimeTime to know which cells can be swapped relative to one another.

Cells That Are Not Down-Sized or Swapped

The area/power recovery process honors any **dont_touch** and **dont_use** attributes that are applied to the design. Just as with Design Compiler and IC Compiler, a **dont_touch** attribute can be applied to an upper-level hierarchical cell, and its implicit effect propagates downward into the hierarchy. If a different **true** or **false** **dont_touch** value is applied deeper in the hierarchy, it takes precedence over any higher-level **dont_touch** attributes. Use **set_dont_use** command to apply the **dont_use** attribute to certain library cells so that they are not used during area/power recovery. The recovery process also honors the **pt_dont_use** user-defined attribute as the **fix_eco_timing** command.

The recovery process does not downsize or swap any cells in the clock network.

In distributed multi-scenario analysis fixing, a restriction in one scenario also should be applied to other scenarios. Any **dont_touch** and **dont_use** attributes should be applied to all scenarios to prevent design changes across all scenarios for the affected cells and nets.

Unconstrained Cells

You can control downsizing or swapping unconstrained cells by specifying the **eco_power_exclude_unconstrained_cells** variable. If the variable is set to **false** (the

default), PrimeTime downsizes or swaps unconstrained cells. If the variable is set to **true**, PrimeTime does not touch unconstrained cells. The variable must be set in the master in the distributed multi-scenario analysis flow.

In distributed multi-scenario analysis flow, a cell is unconstrained if the cell is not constrained in all scenarios. If a cell is not constrained in one scenario but constrained in another scenario, the cell is considered constrained in distributed multi-scenario analysis flow.

UPF

If a cell has the **always_on**, **is_isolation**, **is_retention**, or **is_level_shifter** low-power control attribute set to **true**, it is not considered for cell downsizing or swapping.

Honoring Footprint Attributes in the Library for Cell Swapping

If you want PrimeTime to honor a footprint attribute from libraries, you need to import the attribute and specify the attribute name in the **eco_alternative_cell_attribute_restrictions** variable. Suppose your library has the **cell_footprint** attribute to indicate cell footprint. Then, the following commands restrict the **fix_eco_power** command to swap only footprint-compatible cells.

```
# Import 'cell_footprint' attribute to PrimeTime
define_user_attribute cell_footprint -class lib_cell -import -type string

# Restrict compatible cells only to the same cell_footprint
set eco_alternative_cell_attribute_restrictions {cell_footprint}

# Swappable cells should satisfy both the pattern priority and
# the cell_footprint attribute value
fix_eco_power -pattern_priority {HVT NVT LVT}
```

Restricting Library Cells for Cell Down-Sizing

If you want PrimeTime to exclude certain library cells during downsizing, set their **dont_use** attribute to **true** by using **set_dont_use** command. If you want PrimeTime to restrict the choice of library cells during downsizing, e.g. a normal buffer must not be sized to clock buffers, you need to specify the user-defined attribute name in the **eco_alternative_cell_attribute_restrictions** variable.

For example, suppose that there are six buffers: BUF1X, BUF2X, BUF3X, CLK_BUFX1, CLK_BUFX2, and CLK_BUFX3. If you want BUFX3 to be downsized to either BUFX1 or BUFX2, but not to CLK_BUFX1, CLK_BUFX2, or CLK_BUFX3 (or vice versa), use the following commands:

```
# Define a user-defined attribute 'eco_group" for lib cell.
# Assign different value to each library cell with "good ok bad"
define_user_attribute eco_group -type string -class lib_cell
set_user_attribute -class lib_cell [get_lib_cell BUF1X]      eco_group "buf1"
set_user_attribute -class lib_cell [get_lib_cell BUF2X]      eco_group "buf1"
set_user_attribute -class lib_cell [get_lib_cell BUF3X]      eco_group "buf1"
```

```

set_user_attribute -class lib_cell [get_lib_cell CLK_BUFX] eco_group "buf2"
set_user_attribute -class lib_cell [get_lib_cell CLK_BUFX] eco_group "buf2"
set_user_attribute -class lib_cell [get_lib_cell CLK_BUFX] eco_group "buf2"

# Restrict compatible cells only to the same eco_group
set eco_alternative_cell_attribute_restrictions {eco_group}

# Perform cell downsizing
fix_eco_power

```

Fixing with Multiply Instantiated Modules (MIM)

When the **eco_enable_mim** variable is true, PrimeTime automatically derives the MIM configuration. The MIM configuration is derived from parasitics files. If multiple instances share the same parasitics file using the **-path** option in the **read_parasitics** command, PrimeTime recognizes them as MIM instances.

For more details, see the man page for the **fix_eco_timing** command.

EXAMPLES

The following example shows how to recover area/power by downsizing cells. Both sequential and combinational cells are downsized.

```
pt_shell> fix_eco_power
```

The following example shows how to downsize only combinational cells.

```
pt_shell> fix_eco_power -cell_type combinational
```

The following example shows how to recover leakage with the library cells having prefix HVT, NVT, and LVT for their names. Both sequential and combinational cells are swapped.

```
pt_shell> fix_eco_power -pattern_priority {HVT NVT LVT}
```

The following example shows a typical example for the area/power recovery in distributed multi-scenario analysis.

```

-----
# Update timing & global reporting in master
-----
remote_execute {
    set timing_save_pin_arrival_and_slack true; update_timing -full
}
report_global_timing
report_constraint -all -max_cap -max_transition

-----
# Before area/power recovery:
#   Report timing, power, cell usages for each slave
-----
remote_execute {

```

```

report_global_timing
report_constraint -all -max_cap -max_trans
set power_enable_analysis true
report_power -group {...}
report_cell_usage
}

#-----
# Before area/power recovery:
#   Check compatible library cells for ECO.
#-----
remote_exec {
    report_eco_library_cells
}

#-----
# Execute signoff-driven area/power recovery.
#-----
fix_eco_power -verbose

#-----
# Reporting after area/power recovery
#-----
report_global_timing
report_constraint -all -max_cap -max_transition
remote_execute {
    report_global_timing
    report_constraint -all -max_cap -max_trans
    set power_enable_analysis true
    report_power -group {...}
    report_cell_usage
}
exit

```

SEE ALSO

```

report_cell_usage(2)
report_eco_library_cells(2)
set_dont_use(2)
eco_alternative_area_ratio_threshold(3)
eco_power_exclude_unconstrained_cells(3)
timing_save_pin_arrival_and_slack(3)

```

fix_eco_timing

Fixes or improves timing violations through engineering change order (ECO) changes.

SYNTAX

```
string fix_eco_timing
[-type setup | hold]
[-methods method_list]
[-slack_lesser_than slack_limit]
[-slack_greater_than slack_limit]
[-group group_name]
[-pba_mode none | path | exhaustive]
[-from from_list]
[-to to_list]
[-setup_margin margin]
[-hold_margin margin]
[-buffer_list buffer_list]
[-verbose]
[-current_library]
[-ignore_drc]
[-cell_type combinational | sequential]
[-physical_mode none | open_site | occupied_site]
[-path_selection_options option_string]
```

Data Types

method_list	list
slack_limit	float
group_name	string
from_list	list
to_list	list
margin	float
buffer_list	list
option_string	string

ARGUMENTS

-type setup | hold
Specifies the setup or hold fixing.

-methods method_list
Specifies a fixing method type. Available values are **size_cell**, **insert_buffer_at_load_pins**, and **insert_buffer**.
In hold fixing, you can specify either one or multiple of these methods. The default for hold fixing is to use **{size_cell insert_buffer}**. PrimeTime inserts buffers at only load pins if you specify **insert_buffer_at_load_pins**. PrimeTime inserts buffers at both driver and load pins if you specify **insert_buffer**. Specifying **{insert_buffer_at_load_pins insert_buffer}** is as same as specifying **insert_buffer**.
In setup fixing, you can specify either one of **size_cell** or **insert_buffer** methods but not both of them together in the same command. It is recommended to use **size_cell** first as the primary method for setup fixing. **insert_buffer**

should be used after **size_cell** to gain incremental improvements. The default for setup fixing is to use **{size_cell}** only. Additionally, the **insert_buffer** method can be used in conjunction with **-physical_mode open_site | occupied_site** options only.

-slack_lesser_than slack_limit
 Specifies that only those paths with a slack less than the *slack_limit* option are to be fixed. You can combine this option with the **-slack_greater_than** option to fix paths inside or outside a given slack range. If you do not specify this option, the default **-slack_lesser_than** option limit is zero.

-slack_greater_than slack_limit
 Specifies that only those paths with a slack greater than the *slack_limit* option are to be fixed. This can be useful to avoid spending time on timing paths that are violating due to design problems or incorrect constraints. If you do not specify this option, there is no **-slack_greater_than** option limit.

-group group_name
 Restricts the fixing process to paths in this *group_name* type. Paths are grouped by using the **group_path** or **create_clock** command. You can supply multiple path groups to this option using a list of group names (wildcards are allowed) or a collection of *path_group* objects.

-pba_mode none | path | exhaustive
 Controls the use of path-based analysis. The effort value can be either **none**, **path**, or **exhaustive**, and the meanings are the same as in the **report_timing** and **get_timing_paths** commands. That is, when you specify **none** (the default), path-based analysis is not applied. When you specify **path**, path-based analysis is applied to paths after they are gathered. When you specify **exhaustive**, an exhaustive path-based analysis path search algorithm is applied to determine the worst path set in the design. When specified, the fixing process uses the specified path-based analysis type to determine the timing fixes needed.

-from from_list
 Specifies a list of from pins, ports, or nets to be used for timing path fixing. This option is consistent with the **report_timing** and **get_timing_paths** commands. The default behavior is to select the path with the worst slack within each path group if the design has timing constraints.

-to to_list
 Specifies a list of to pins, ports, or nets to be used for timing path fixing. This option is consistent with the **report_timing** and **get_timing_paths** commands. The default behavior is to report the path with the worst slack within each path group if the design has timing constraints.

-setup_margin margin
 Specifies an additional fixing margin to be applied to setup slacks. By default, the setup margin is zero. During setup fixing, a positive value specifies overfixing by that amount, and a negative value specifies underfixing by that amount. The value is specified in a library unit. During hold fixing, a positive value means that the specified amount of setup slack would be preserved, and a negative value means that specified amount of setup slack would be compromised to fix hold violations.

-hold_margin margin
 Specifies an additional fixing margin to be applied to hold slacks. By default, the hold margin is zero. During hold fixing, a positive value specifies overfixing by that amount, and a negative value specifies underfixing by that amount. The value is specified in a library unit. During setup fixing, a positive value means that the specified amount of hold slack would be preserved, and a negative value means that specified amount of hold slack would be compromised to fix setup violations.

-buffer_list buffer_list
 Specifies the list of allowable buffers to use for setup or hold fixing with **insert_buffer** method. Buffer names must be in a base_name form. Library name is resolved by the specified link path as the **insert_buffer** command does. This option is required when performing setup or hold fixing with the **insert_buffer** method. If a library cell is specified by this option, PrimeTime uses the library cell for setup or hold fixing, even if it has the **dont_use** attribute; this is to honor the intention of the user who triggered the command. In hold fixing, if you use the **-methods {size_cell insert_buffer}** option, the tool may size the inserted buffers to buffers that are not specified by the **-buffer_list** option.

-verbose
 Shows additional information during the fixing process. In distributed multi-scenario analysis, violation information from each scenario is displayed with this option. Upon completion, it shows unfixable reasons for those that are not fixed if the variable **eco_report_unfixed_reason_max_endpoints** is set to a positive integer. The report also shows fixing priority, ranging from P0 (lowest fixing priority) to P9 (highest fixing priority). The priority indicates the effectiveness of fixing the cell. For example, fixing a cell with P9 priority would potentially fix more endpoint violations than fixing a cell with P1 priority.

-current_library
 Specifies to use library cells only from the current library that the cell belongs to when **size_cell** is specified for the **-methods** option. This can improve runtime but might degrade quality of results, as the number of available alternative library cells for sizing is reduced.

-ignore_drc
 By default, PrimeTime does not fix a timing violation when there are any DRC violations on the path. When you specify this option, PrimeTime ignores the max_transition, max_capacitance, and max_fanout DRC violations and prioritizes the fixing of hold or setup violations. Note that using the **-ignore_drc** option might degrade DRC violations.

-cell_type combinational | sequential
 Specifies cell type for fixing. The cell type can be either **combinational** (the default) or **sequential**. When **combinational** is specified, fixing is only allowed in data path. When **sequential** is specified, sequential cells are resized to fix violations at its input or output pin. Note that changing sequential cells can affect clock tree timing, so this capability should only be used when it is absolutely necessary.

-physical_mode none | open_site | occupied_site
 Specifies the use of physical data to resize cells or place buffers.

- **none** (the default) - Does not use physical data while fixing violations.
- **open_site** - Sizes a cell if there is enough room available around the cell and inserts a buffer if there is an empty site. This mode avoids moving existing cells; it places cells only in empty sites.
- **occupied_site** - PrimeTime considers layout density in the cell neighborhood and can place or resize a cell overlapping existing neighbor cells when the local placement density can accommodate the change. This mode places or resizes a cell even if no empty site is available; to implement the change list, you need a place-and-route tool to move existing cells to create a room for the new or sized cell.

-path_selection_options option_string

Specifies the string of path selection options for the **get_timing_paths** command. The tool uses these options to collect timing paths to target fixing. Since the **-path_selection_options** option allows fixing on custom selection of paths, the quality of result should be gauged by relevant reports (for example, **get_timing_paths** or **report_timing** command with the identical options used for ECO). The **-path_selection_options** option cannot be used together with the **-from**, **-to**, **-group**, **-pba_mode**, **-slack_lesser_than**, or **-slack_greater_than** option of the **fix_eco_timing** command. The **-path_selection_options** option

- Supports options of the **get_timing_paths** command that are relevant to ECO. Using unsupported options results in errors.
- Must be enclosed with curly braces { } when it contains any collection.
- Must have the **-max_paths** and **-nworst** options explicitly specified with values that are suitable for ECO unless either the **-start_end_pair** or **-cover_design** option is specified.

DESCRIPTION

This command seeks to make ECO changes to the design that improves or resolves setup or hold violations. You can use the **fix_eco_timing** command in the single-core analysis, distributed multi-scenario analysis, and multicore analysis flows within PrimeTime. It uses algorithms that attempt to improve the specified slack type, while minimizing the number of ECO changes as well as the area impact of the changes.

You must specify a **setup** or **hold** fixing type with the required **-type** option. The fixing strategies are different for setup and hold fixing, due to the different nature of the violations. Setup fixing uses cell resizing to reduce logic delay and buffer insertion to reduce path delay to improve setup slack. Hold fixing uses buffer insertion to add delay and improve hold slack.

The fixing process honors any **dont_touch** and **dont_use** attributes that are applied to the design. For more information, see the "Restricting the Selection of Cells Used for Resizing" and "Preventing Fixes on Specific Cells or Nets" sections.

Neither fixing type makes changes to clock nets used as data where timing violations

exist. Setup fixing seeks to avoid introducing design rule checking (DRC) violations, but it is permitted to introduce hold violations since setup is a harder problem to fix. Hold fixing seeks to avoid introducing both setup and DRC violations. If you want to fix both types, it is recommended that you first fix setup timing and then fix hold timing.

In distributed multi-scenario analysis fixing, a restriction in one scenario also should be applied to other scenarios. For example, **dont_touch** and **dont_use** attributes should be applied to all scenarios to prevent design changes across all scenarios for the affected cells, nets and library cells.

The **-group**, **-from**, and **-to** options allow targeted fixing of the design. The **-group** option allows fixing to be limited to paths ending at violating endpoints that belong to one or more specified path groups. If a violating endpoint belongs to multiple path groups and not all the path groups are specified in **-group** option, path groups that are not specified can also be considered for fixing the specific violating endpoint. You can use lists and wildcards to specify the list of allowable path groups. The **-from** and **-to** options allow fixing only for paths linked to a certain set of startpoints or endpoints, respectively. You can specify both the **-from** and **-to** options to fix timing only along a certain path.

The **fix_eco_timing** command requires that pin slack information be available in the design. If you have not set the **timing_save_pin_arrival_and_slack** variable to **true** (the default is **false**), the **fix_eco_timing** command automatically sets it to **true** and updates the design with arrival and slack information.

Slack threshold controls are available to provide additional control over the fixing process. The **-slack_lesser_than** option indicates that only violations less than this value should be fixed. In addition, you can use the **-slack_greater_than** option to avoid spending time fixing gross timing violations that might be caused by incorrect constraints. The slack thresholds only pertain to the slack of the specified fixing type (setup or hold).

Note that the **fix_eco_timing** command is designed for the sign-off flow with parasitics, in which delays and slews are calculated based on parasitics. If delays and slews are not computed inside PrimeTime but annotated such as in Standard Delay Format (SDF) flow, the quality of results cannot be guaranteed.

Setup Fixing

When cells are enlarged after setup fixing there is no default area limit imposed on the alternative cells considered for resizing. However, if there is a specific requirement on the maximum size increase of a resized cell, you can control the area increase by setting the **eco_alternative_area_ratio_threshold** variable that specifies the maximum area increase as a ratio. By setting this variable to 2, the resizing is limited to twice the area of the original cell. A value of zero (the default value) means that no limit is imposed on upsizing. Imposing an maximum area increase can increase the timing predictability after physical implementation of the ECO changes, as the smaller cell size increases are less likely to perturb layout during incremental placement and routing. However, more changes (and more runtime) might be required to achieve the needed slack improvement. When buffering is desired to reduce path delays to critical loads, it is recommended to use **size_cell** first to reduce the number of buffers inserted to minimize layout disturbance when the changes are applied to IC Compiler. **insert_buffer** should be used after **size_cell** to

gain incremental improvements in setup fixing.

Buffering for setup fixing is only available in conjunction with "**-physical_mode none | open_site | occupied_site**" options. This technique to improve setup fix-rate, is heavily dependent on the physical layout and the empty sites available. You should provide a good representative set of buffer that cover the necessary delay and area range. However delay buffers with large delays and clock buffers should be avoided from this list as they will not be useful in reducing the path delays.

Hold Fixing

For hold fixing, cells are downsized, or buffers are inserted to remove hold violations. Both methods, **size_cell** and **insert_buffer** can be used together or separately. By default, both methods are used. When both **size_cell** and **insert_buffer** are used, **size_cell** is first applied to reduce the number of buffers to minimize layout disturbance when the changes are applied to IC Compiler. If the violation still exists, then, **insert_buffer** is applied.

You should provide a good representative set of buffer and delay cells that covers the necessary delay range. Supplying too many buffers in the list might not improve results and can also lead to longer runtime.

Restricting the Selection of Cells Used for Resizing

To prevent PrimeTime from using specific library cells for resizing, apply the **dont_use** attribute to these cells with the **set_dont_use** command.

However, if you specify library cells in **-buffer_list** option, PrimeTime uses all those specified library cells for setup or hold fixing, regardless of their **dont_use** attribute.

In addition, use of library cells can be also restricted by applying the **pt_dont_use** user-defined attribute on the restricted lib_cell objects, which is a legacy feature. For an example of how to apply this attribute, see the EXAMPLES section.

Preventing Fixes on Specific Cells or Nets

To prevent PrimeTime from performing fixes on specific cells or nets, apply a **dont_touch** attribute to the cell or net. As with Design Compiler and IC Compiler, a **dont_touch** attribute can be applied to an upper-level hierarchical cell, and its implicit effect propagates downward into the hierarchy. If a different **true** or **false** **dont_touch** value is applied deeper in the hierarchy, it takes precedence over any higher-level **dont_touch** attributes. The **dont_touch** attributes can also be placed on specific cell and net objects.

During setup fixing, if a cell has a **dont_touch** setting, which was applied directly or at a higher hierarchical level, PrimeTime does not resize the cell. Additionally, if a net segment directly connected to the pin has a **dont_touch** setting, which was applied directly or at a higher hierarchical level, PrimeTime does not insert buffers at that net.

During hold fixing, if a net segment directly connected to the pin has a **dont_touch**

setting, which was applied directly or at a higher hierarchical level, PrimeTime does not insert buffers at that net.

Fixing Margin

Fixing margins are also available to provide additional control over the setup and hold fixing process. For example, a positive **-hold_margin** value during hold fixing indicates that a hold violation should be overfixed by the specified amount. This extra margin can be useful to account for timing differences between the predicted timing and the actual layout timing in PrimeTime due to scenic routes, legalization effects, and so on. In hold fixing, you can use the **-setup_margin** option to control how setup slack is preserved. By setting a positive **-setup_margin** value during hold fixing, the hold fixing process attempts to preserve that much positive setup slack while fixing hold violations. By specifying a negative **-setup_margin** value, hold fixing limits its setup slack impact to that negative value. Note that the **-hold_margin** value does not affect which hold violations are fixed. The **-slack_lesser_than** is used for this purpose. However, the **-hold_margin** value does provide a way to specify how much "overfixing" should be performed on those violations.

When the **-pba_mode** option is specified with either the **path** or **exhaustive** value, the margin is applied to the graph-based analysis value if the margin is not in the same fixing type. For example, in hold fixing with path-based analysis, the **-setup_margin** option controls whether to preserve the graph-based analysis setup slack while the **-hold_margin** value is relative to the hold path-based analysis.

Fixing Priority

Even though it is recommended that you first fix setup timing and then fix hold timing, you might want to fix hold timing first followed by setup fixing. In such cases, you can specify a positive **-hold_margin** value to preserve hold slacks during setup fixing. Moreover, if additional hold violations are introduced, you might want to run the **fix_eco_timing** command again to fix them.

Fix Rate

The **fix_eco_timing** calculates its fix rate by the percentage of violating endpoints that are fixed after the command is executed. Note that it is not the number of violations but the number of violating endpoints, which means rise and fall violations for the same endpoint are counted as one. For example, suppose there are 100 violating endpoints and 99 of them are fixed. Then, the fix rate is 99%.

In distributed multi-scenario analysis, fix rate is calculated as the percentage of the total sum of fixed violating endpoints across all scenarios. For example, suppose there are 100 violating endpoints in the scenario 1 and 200 violating endpoints in the scenario 2. The **fix_eco_timing** command fixes 99 violations in scenario 1 and 198 in scenario2. Then, fix rate is $(99 + 198) / (100 + 200) = 99\%$.

Remaining Violations

The **fix_eco_timing** command applies an intelligent adaptive algorithm to optimize runtime, memory, and the quality of result (QoR), such as fix rate and the number of

changes. Thus, it is possible that some of violations are still remaining after fixing if it is determined that spending more time to fix violations would not produce better results. For example, if a path has a huge negative hold violation slack and requires 10 or 20 buffers to be inserted in one stage, the **fix_eco_timing** command stops fixing the path. Due to the lack of physical information in PrimeTime, if a large number of buffers are inserted to one stage, the timing difference between PrimeTime and the implementation tool such as IC Compiler can be large. In this case, it is recommended to fix such violations in the implementation tool or adjust the constraint so that violations are realistic.

However, if you still want to fix those violations, you can always run the **fix_eco_timing** command multiple times until they are all fixed.

UPF

If a cell has the **always_on**, **is_isolation**, **is_retention**, or **is_level_shifter** low-power control attribute set to **true**, it is not considered for resizing. Buffer insertion is not allowed for the net connected to all pins of an **always_on** cell, control and clock pins of a retention cell. Buffer insertion is also prohibited for the nets between a port and an isolation cell or between a port and a level shifter cell.

Reporting Changes

PrimeTime reports the number of resized cells and inserted buffers when it completes fixing timing violations. The numbers may not match the number of changes written by the **write_changes** command because the **write_changes** command optimizes the changes. For example, suppose the **fix_eco_timing** command inserts one buffer with the lib cell BUF1X in the first iteration and then resizes the buffer to the next size lib cell BUF2X in the second iteration. PrimeTime reports one buffer insertion and one resizing. However, the **write_changes** command detects the resizing has been done on the inserted buffer and writes only one **insert_buffer** command but with the final lib cell, BUF2X. Another example is the **add_buffer_on_route** command that can merge multiple buffers in one net into only one command.

Unfixable Violations

If you specify **fix_eco_timing** with the **-verbose** option, and set the **eco_report_unfixed_reason_max_endpoints** variable with a positive integer, the tool reports the unfixed violations with the following explanations. For more information, run the **estimate_eco** command on specific cells to see the details of estimated timing violations, delays, and slacks.

- **A - There are available library cells outside area limit**

This reason appears if you set **eco_alternative_area_ratio_threshold** variable to limit the area of alternative library cells, and there are other alternative library cells with larger area, which could be considered for fixing. You might be able to fix the violation by increasing the area limit.

- **B - Delay improvement is too small to fix the violation**

This reason appears if the violating slack is too large to be fixed by improving the delay in this stage. Use margin to reduce the target. For example, for hold fixing, if the violation has -100 ps slack, use **-hold_margin -80** ps so that the target delay improvement is reduced from 100 ps to 20 ps.

- **C - The violation is in clock network**

This reason appears if the violation is in a clock network. Any violation in a clock network is not fixed because it can change the clock tree timing and potentially create new timing violations.

- **D - Cell or net is located in high density area**

This reason appears if there is not enough room available in the vicinity of the stage. Inspect your LEF/DEF file for filler cells and validate whether they can be and are treated as open_sites. When the eco_allow_filler_cells_as_open_sites variable is set to true, physically-aware ECO commands treat all filler cells as open_sites. This variable is by DEFAULT set to true. Additionally, place-and-route tool can also be used to reduce the region density.

- **E - Physical information is incomplete or unavailable**

This reason appears if physical data is not available or incomplete for the specific net or cell. Check if the DEF and LEF files are consistent with your current design.

- **H - Logical and physical hierarchy is inconsistent**

This reason appears if the netlist and physical layout are not consistent, and a buffer cannot be inserted. Use place-and-route tool to resolve inconsistency.

- **I - Buffer insertion with given library cells cannot fix the violation**

This reason appears if the violation cannot be fixed with given buffer library cells specified by the **-buffer_list** option. Add more buffer library cells, if needed.

- **L - Available physical area limits the use of one or more library cells**

This reason appears if the library cell's area exceeds the available physical area. Use place-and-route tool to increase the available physical area.

- **O - No open free site is available**

This reason indicates that there is no empty free site available on the route or in the vicinity of the specified pin or cell. Inspect your LEF/DEF file for filler cells and validate whether they can be and are treated as open_sites. When the eco_allow_filler_cells_as_open_sites variable is set to true, physically-aware ECO commands treat all filler cells as open_sites. This variable is by DEFAULT set to true. Additionally, place-and-route tool can also be used to create free sites.

- **S - Cell sizing with alternative library cells cannot fix the violation**

This reason indicates that all alternative library cells have been considered for resizing to fix the violation, but the violation cannot be fixed. Use a buffer insertion method to fix the violation, if needed.

- **T - Timing margin is too tight to fix the violation**

This reason indicates that the violation is on a critical or near-critical setup or hold timing path and does not have enough timing slack to fix the violation. You can relax this by providing more timing margins and specifying negative values for the **-setup_margin** and **-hold_margin** options.

- **U - UPF restricts fixing the violation**

This reason indicates that certain UPF cells cannot be sized or buffered. See the UPF section of this man page for the details of UPF cells.

- **V - Net or cell is invalid or has dont_touch attribute**

This reason appears if the leaf cells and nets under the cell has the **dont_touch** attribute set to **true**. Check if invalid reasons are justified.

- **W - Fixing the violations might degrade DRC violations**

This reason appears if the delay fixing create new DRC violation or worsen the existing DRC violations. Use the **-ignore_drc** option if fixing violation is preferred at the expense of DRC degradation.

The following example shows the output example of remaining unfixed violations. The Reasons column shows one or multiple reasons for the stage. The Prio/slk column shows an endpoint slack or fixing priority starting from the lowest priority P0 to the highest priority P9.

```
pt_shell> fix_eco_timing -type hold -buffer_list { BUFX1 BUFX2 BUFX3 } \
           -verbose -physical_mode open_site
```

Unfixable violations:

- A - There are available library cells outside area limit
- B - Delay improvement is too small to fix the violation
- C - The violation is in clock network
- D - Cell or net is located in hight density area
- E - Physical information is incomplete or unavailable
- H - Logical and physical hierarchies are inconsistent
- I - Buffer insertion with given library cells cannot fix the violation
- L - Available physical area limits the use of one or more library cells
- O - No open free site is available
- S - Cell sizing with alternative library cells cannot fix the violation
- T - Timing margin is too tight to fix the violation
- U - UPF restricts fixing the violation
- V - Net or cell is invalid or has dont_touch attribute
- W - Fixing the violation might degrade DRC violations

Violation	Reasons	Prio/slk

S:U2/Z	T	P7
U3/Z	I W	P6
U4/Z	U	P9
U6/Z	U	P6
E:U8/I		-2.105
C:U3/Z	I W	P6
U5/Z	S	P3

U7/Z
E:U9/A

U P0
-0.307

The unfixable reason "T" shows that U2 was not fixed because the timing margin is too tight. "P7" shows that the priority is P7 which range from P0 to P9. The first segment starts with "S" and ends with "E" indicating the start pin and the end pin of the segment. Since it is the worst path, the segment is a full path starting from U2 ending at U8. The second segment is reported starting from U3 ending at U9 omitting the segment of the path from U2 and U3. This avoids duplicate showing the unfixable reason for common path. The second segment starting with "C" indicates that U3 is a connect pin not the start pin of a segment.

Note: Timing endpoints marked by "E" are sorted by slack values with the worst slack at the top of the table. Therefore, segments at the top of the table are more critical than segments at the bottom of the table. Within a segment, the priority column shows the fixing priority. For example, in the first segment in the table above, fixing the stage with U3/Z (priority P9) is potentially more effective than fixing the stage with U6/Z (priority P6).

You can also view the fixing priority as timing fixing bottleneck. If you have a large number of unfixed violations but want to fix only some of them, you can collect stages with higher priorities and fix those stages first to fix more violations.

Fixing Multiple Scenarios With Fewer Hosts

PrimeTime SI requires the same number of host as the number of scenarios because the best fixing rate and fastest runtime can be achieved when the same number of hosts are used. It maximizes parallelism with distributed multi-scenario processing. However, in some cases, you might need to perform ECO fixing on fewer hosts. The **fix_eco_timing** command allows this configuration if the **eco_enable_more_scenarios_than_hosts** variable is set to **true**. Note that this setting can affect performance and QoR as the number of hosts is reduced, and the **fix_eco_timing** command runs in a special mode to optimize computing resource, runtime and QoR.

If fewer hosts are available than scenarios, the hosts run in scenario swapping mode in which they rotate and swap scenarios one after the other. Thus, it can significantly degrade performance.

In order to achieve faster runtime, it is recommended to minimize the number of merged reporting commands such as **report_timing** and **report_analysis_coverage** as they trigger a full round of scenario swapping. Moreover, reduce the number of **remote_exec{}** commands and merge them into one **remote_exec{}** block. For example, the following script triggers four rounds of scenario swapping:

```
remote_exec { set_false_path ... }  
remote_exec { set_false_path ... }  
remote_exec { set_dont_touch ... }  
remote_exec { set_dont_touch ... }
```

In the following example, the commands are merged into one block, which triggers only one round of scenario swapping:

```

remote_exec {
    set_false_path ...
    set_false_path ...
    set_dont_touch ...
    set_dont_touch ...
}

```

Ignoring DRC Violations

If you specify the **-ignore_drc** option, PrimeTime ignores DRC violations and tries to fix more hold or setup violations. This option is not recommended for the early stages of an ECO cycle. During the later stages of an ECO cycle, when no more timing violations can be fixed because of DRC violations, you can use this option to trade off timing violations at the expense of DRC violations.

Working with the Place and Route Tool

After PrimeTime completes the **fix_eco_timing** command, use the **write_changes** command to create a change list that can be implemented in your place and route tool, such as IC Compiler. Then, extract new parasitic data, write a new Verilog file, and run PrimeTime again to check the final timing. After you use the **fix_eco_timing** command, do not use the **read_parasitics** command to incrementally add changed parasitics because your changed design in PrimeTime might be different from the design generated by the place and route tool.

Physically-Aware Timing Fixing

Specifying **open_site** or **occupied_site** for the **-physical_mode** option enables PrimeTime to load physical data into memory and use them to fix setup or hold timing violations. If you want to minimize layout disturbance for your ECO changes, such as in the late stage of your ECO cycle, use the **open_site** mode not to move existing cells in the layout. On the other hand, if you want to maximize your fix rate, but it is allowed to move existing cells, such as in the early stage of your ECO cycle, use the **occupied_site** mode in which PrimeTime considers layout density and attempts to maximize fix rate by placing cells even if enough free size is unavailable.

Before running the **fix_eco_timing** command, use the **set_eco_options** command to specify the paths to the physical data, such as Design Exchange Format (DEF) files and Library Exchange Format (LEF) files. PrimeTime reads the physical data during the **fix_eco_timing** command.

With the physical data loaded into memory, PrimeTime considers physical constraints such as available free site, cell density and net topology when it resizes cells and inserts buffers. For resizing, PrimeTime selects a lib cell that meets both timing and physical constraint. For buffering, PrimeTime searches for the best placement location that maximizes timing improvement but minimizes layout disturbance.

After the **fix_eco_timing** command is completed, you can write a change list that can be used by IC Compiler to implement the ECO changes. With the physical mode enabled, the commands in the change list contain placement information. The following example shows the **add_buffer_on_route** command that specifies the layout coordinates, X = 100.0 and Y = 200.0, for the new buffer. For details about the **add_buffer_on_route**

command, see the IC Compiler man pages.

```
add_buffer_on_route net1 BUF1X -location {100.0 200.0 0}
```

The following example shows the `size_cell` command selecting the new lib cell that fits in the available space around the cell. Note that it does not specify any location because the cell is upsized in the current location.

```
size_cell U1 AND2X
```

The following example shows the `insert_buffer` command to place the new buffer in an available free site at X = 200.0 and Y = 300.0.

```
insert_buffer U2/Z BUF2X -location {200.0 300.0 0}
```

Note: When physical ECO is enabled, you can specify only the icctcl format in the `write_changes` command.

Physically-Aware Timing Fixing and Other Fixing

After physical data is loaded into memory, PrimeTime keeps track of changed cells and inserted buffers and updates the physical database with the changes. Therefore, it is not recommended to mix physically-aware fixing and non-physically-aware fixing in a single PrimeTime session. For example, do not run the `fix_eco_timing` command with the `-physical_mode` option set to `none` if you have already performed timing or DRC fixing with the `-physical_mode` option set to `open_site` or `occupied_site`. Otherwise, the changes in non-physically-aware fixing can corrupt the physical database and results in significant layout disturbance while they are implemented in IC Compiler. After the physical data is loaded into memory , it is also not recommended to use what-if commands such as the `size_cell` and `insert_buffer` because they are not physically-aware.

Fixing with Multiply Instantiated Modules (MIM)

When the `eco_enable_mim` variable is true, PrimeTime automatically derives the MIM configuration. In physically-aware fixing, the MIM configuration is derived from DEF files. If multiple instances share the same DEF file, PrimeTime recognizes them as MIM instances. In non-physically-aware fixing, MIM configuration is derived from parasitics files. If multiple instances share the same parasitics file using the `-path` option in the `read_parasitics` command, PrimeTime recognizes them as MIM instances.

Suppose CPU module is instantiated four times in TOP module as CPU1, CPU2, CPU3 and CPU4, and there are CPU.def and CPU.SPEF associated with CPU module. PrimeTime derives its MIM configuration when the following setting is specified: In physically-aware fixing,

```
set_eco_options -physical_design_path {TOP.def CPU.def}
```

In non-physically-aware fixing,

```
read_parasitics -path {CPU1 CPU2 CPU3 CPU4} CPU.SPEF
```

Alternatively,

```
read_parasitics -path [all_instance CPU] CPU.SPEF
```

If MIMs are detected in the design, the **fix_eco_timing** command shows the current MIM configuration before starting fixing. Verify that the MIM configuration matches your expectation. If not, check the DEF file or parasitics file configuration in your script.

When fixing violations, if a change occurs in one of the MIM instances, PrimeTime replicates the change to all MIM instances. In the preceding example, a buffer insertion in CPU1 is replicated in CPU2, CPU3, and CPU4. Whenever PrimeTime fixes violations, it analyzes all MIM instances and makes sure that it does not break timing in other MIM instances. Thus, it is possible to see lower fix rate with MIM enabled because timing might be more constrained by multiple instances. However, it is the most realistic timing that matches the timing after the changes are implemented in the place-and-route tool.

After fixing is completed, use the **write_changes** command to generate one change list file for the associated MIM instances. In the preceding example, one change list file is written for both CPU1 and CPU2 instances. For more information, see the man page of the **write_changes** command.

EXAMPLES

The following example shows how to fix setup violations less than zero slack, but avoiding some largely violating setup paths with slack less than -2:

```
pt_shell> fix_eco_timing -type setup -slack_greater_than -2
```

The following example shows how to overfix setup violations less than zero slack to achieve setup slack of 2 while preserving hold slack of 1:

```
pt_shell> fix_eco_timing -type setup -setup_margin 2 -hold_margin 1
```

The following example shows how to overfix setup violations with a slack less than -1 to achieve setup slack of 1:

```
pt_shell> fix_eco_timing -type setup -slack_lesser_than -1 -setup_margin 1
```

The following example shows how to use only sizing method to fix hold violations.

```
pt_shell> fix_eco_timing -type hold -methods size_cell
```

The following example shows how to fix a small set of hold violations that have an exhaustive path-based slack less than zero:

```
pt_shell> fix_eco_timing -type hold -pba_mode exhaustive \
          -buffer_list {BUFX2 DLY1X2 DLY2X2}
```

The following example shows how to fix setup violations only in a selected set of path groups, with verbose output enabled:

```
pt_shell> fix_eco_timing -type setup -group {SYSCLK* IOCLK} -verbose
```

The following example shows how to fix setup violations by sizing sequential cells:

```
pt_shell> fix_eco_timing -type setup -cell_type sequential
```

The following example shows how to apply the **pt_dont_use** user-defined attribute to a set of library cells. You must define the user-defined attribute first:

```
pt_shell> define_user_attribute pt_dont_use \
           -quiet -type boolean -class lib_cell
```

You can use the attribute directly with the **set_user_attribute** command, or, for convenience, you can define the following procedure:

```
proc set_pt_dont_use {lib_cell} {
    set_user_attribute \
        -class lib_cell \
        [get_lib_cell -quiet $lib_cell] \
        pt_dont_use true
}
```

Now, you can use this procedure to apply the attribute:

```
set_pt_dont_use {lib/CLKBUF* lib/CLKMUX*}
```

In the distributed multi-scenario analysis flow, you must define and apply the attribute at the scenarios using the **remote_execute** command.

The following example shows how to use the **-physical_mode** option to specify the **open_site** mode.

```
pt_shell> fix_eco_timing -type hold -physical_mode open_site \
           -buffer_list {BUFX2 DLY1X2 DLY2X2}
```

The following example shows how to use the **-path_selection_options** option to target specific paths for fixing setup violations.

```
pt_shell> fix_eco_timing -type setup -path_selection_options \
           {-through u01/A -to ff0/D -max_paths 10 -nworst 5}
```

The following example shows how to use the **-path_selection_options** option to target specific paths for fixing hold violations.

```
pt_shell> fix_eco_timing -type hold -buffer_list {BUFX2 DLY1X2 DLY2X2} \
           -path_selection_options {-delay_type min -from ff0/Q -max_paths 100 -nworst 10}
```

In the distributed multi-scenario analysis flow, you must define variables in the slaves if you want to use them in the **-path_selection_options** option. The following example shows how to use the **-path_selection_options** option in the distributed multi-scenario analysis with a variable.

```
remote_exec {
    set startpoints "ff0/Q ff1/Q"
}
fix_eco_timing -type hold -buffer_list {BUFX2 DLY1X2 DLY2X2} \
```

```
-path_selection_options {-delay_type min -from $startpoints -max_paths 100 -  
nworst 5}
```

SEE ALSO

```
create_clock(2)  
get_timing_paths(2)  
group_path(2)  
remote_execute(2)  
report_timing(2)  
set_dont_use(2)  
set_eco_options(2)  
set_user_attribute(2)  
write_changes(2)  
eco_alternative_area_ratio_threshold(3)  
eco_report_unfixed_reason_max_endpoints(3)  
timing_save_pin_arrival_and_slack(3)
```

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

```
string foreach_in_collection
itr_var
collections
body
```

Data Types

<i>itr_var</i>	string
<i>collections</i>	list
<i>body</i>	string

ARGUMENTS

<i>itr_var</i>	Specifies the name of the iterator variable.
<i>collections</i>	Specifies a list of collections over which to iterate.
<i>body</i>	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\(2\)](#)

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

```
string get_alternative_lib_cells
[-current_library]
[-libraries lib_spec]
[-base_names]
object
```

Data Types

object	string
lib_spec	list

ARGUMENTS

object
Specifies a cell or a library cell for which alternative library cells is required.

-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 may 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.

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.

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_per_instance(3)`
`link_path(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

<i>class_name</i>	string
<i>object_spec</i>	string or collection
<i>attr_name</i>	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 unavailability 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]
[patterns | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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]

```

Data Types

<i>expression</i>	string
<i>patterns</i>	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_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
```

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 for UPF mode

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
```

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 for UPF mode

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_attributes

Returns attribute names currently defined for the specified class.

SYNTAX

```
string get_defined_attributes
-class class_name
-return_classes
[-details]
[-application]
[-user]
[attr_pattern]
```

Data Types

<i>class_name</i>	string
<i>attribute_names</i>	list
<i>data_type</i>	string

ARGUMENTS

```
-class class_name
    Specifies the class for which to get the attributes. Valid classes are
    application defined. Note legal with the -return_classes option.

-return_classes
    Return the set of application defined classes. Not legal with the -class
    option.

-details
    Gets additional information on a single attribute.

-application
    Returns application-defined attributes only. This option is mutually
    exclusive with the -user option.

-user
    Returns user-defined attributes only. This option is mutually exclusive with
    the -application option.

attr_patterns
    Specifies a list of attribute names or glob style patterns to check on the
    existences of.
```

DESCRIPTION

This command returns a list of attribute names that are defined for the specified class.

There are three legal forms for this command:

- `get_defined_attributes -return_classes`

Returns the application defined object classes

- `get_defined_attributes -class cell [<pattern>]` Returns all the attributes matching the optional pattern

- `get_defined_attributes -class cell -details <attrName>` Returns more detailed information about the attribute. See below 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 attribute

`infor`

The short help defined for the attribute

`type`

The type of value the attribute storesn

`settable`

The value is settable with the `set_attribute` command.

`application`

The value is 1 if the attribute is application defined, else 0.

`subscripted`

The value is 1 if the attribute is subscripted, else 0.

`min_value`

Not always present. If present value is the minimum allowed value.

`max_value`

Not always present. If present value is the maximum allowed value.

`allowd_values`

Not always present. If present value is the set of allowed values

`allowed_subscripts`

Present if the attribute is subscripted. Contains list of legal subscripts for the attribute if the attribute uses global subscripts. If empty, then the subscripts vary per object.

`smart_subscripts`

Present if the attribute is subscripted. Contains the list of supported smart subscripts if the attribute supports smart subscripts.

EXAMPLES

The following are examples of the **get_defined_attributes** command:

```
prompt> get_defined_attributes -class cell
name full_name object_class ....

prompt> get_defined_attributes -class cell *name*
name full_name reference_name ...

prompt> get_defined_attributes -class cell name
name name info {The simple name of the cell} type string application 1
settable 0 subscripted 0
```

SEE ALSO

[get_attribute \(2\)](#)
[help_attributes \(2\)](#)

get_defined_commands

Get information on defined commands and groups.

SYNTAX

```
string get_defined_commands [-details] [-groups] [pattern]
stringpattern
```

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 is 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.

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]

```

Data Types

<i>expression</i>	string
<i>patterns</i>	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_distributed_variables

Gets Tcl variables from slaves and creates an array at the master indexed by the scenario name.

SYNTAX

```
status get_distributed_variables
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-attributes attribute_list]
[-merge_type min | max | average | sum | list | unique_list | none]
[-null_merge_method ignore | error | override]
variable_list
```

Data Types

<i>pre_command_string</i>	string
<i>post_command_string</i>	string
<i>attribute_list</i>	list
<i>variable_list</i>	list

ARGUMENTS

-pre_commands *pre_command_string*

Specifies a string of commands to be executed in the slave context before the retrieval of variables. Groups commands with the semicolon (;) character. The maximum length of the command string is 1000 characters.

-post_commands *post_command_string*

Specifies a string of commands to be executed in the slave context after the retrieval of variables. Groups commands with the semicolon (;) character. The maximum length of the command string is 1000 characters.

-attributes *attribute_list*

Specifies a list of attributes to be retrieved from a slave collection. If you do not use this option, only the **full_name**, **scenario_name**, and **object_class** attributes are retrieved. Use this option together with the **set_distributed_parameter -collection_levels** command to control the amount of data retrieved from the slave.

-merge_type min | max | average | sum | list | unique_list | none

Merges variable values that come from the scenarios as they are retrieved. You can specify one of the following arguments:

- **min** - Uses the minimum variable value.
- **max** - Uses the maximum variable value.
- **average** - Uses the average of all variable values.
- **sum** - Uses the sum of all variable values.

- **list** - Generates a list of all variable values.
- **unique_list** - Merges lists of object name lists. This mode is useful when you want to combine multiple lists of design object names, such as cell, pin, or net names, and there might be small variations between the scenario results due to differing constants, constraints, or interface logic model (ILM) netlists, or differences in the design. Where possible, the **unique_list** merging mode keeps the sublists separate so that gaps can be filled, but merges these sublists together as needed. Using this merging mode is similar to applying the list merging type to each of the sublists, but additionally merges together any sublists that share common items.

- **none** (default)- Brings back a scenario variable as an array.

In addition to simple variables, arrays of one or more dimensions can also be merged. The merging of collection objects is not supported.

-null_merge_method ignore | error | override

Specifies the tool behavior when a null (empty) value of {} is encountered during the merging operations. You can specify one of the following arguments:

- **ignore** (default) - Ignores the null value.
- **error** - Issues an error message if a null value is found during a merging operation.
- **override** - Allows a null value to win during a merging operation.

variable_list

Specifies a list of variables.

DESCRIPTION

The **get_distributed_variables** command retrieves data from the slaves and makes it available for further processing at the master. The variables specified in the **variable_list** option must be slave variables. Each slave variable can have a different value depending on the scenario context. The command creates an array at the master for each variable. The array has the same name as the variable. The array is indexed by the scenario name. The data in the array for a given scenario index has the same value as the slave variable in that scenario's context. This command supports the following variable types: Tcl arrays, Tcl lists, and all types of collections.

To merge variable values that come from the scenarios as they are retrieved, use the **-merge_type** and **-null_merge_method** options. This is especially useful when you are writing customized distributed multi-scenario analysis scripts.

By default, the **get_distributed_variables** command brings back a scenario variable as an array. With the **-merge_type** option, the values can be merged back into a variable during retrieval.

When the number of scenarios is greater than the number of processors, variables can be destroyed during the save and restore process. When the number of scenarios is less than the number of processors, use the **pre_commands** variable to generate the

variables to be transmitted. When retrieving a collection, use the **-attributes** option to specify what attributes are to be retrieved from the slaves for the given collection.

EXAMPLES

In the following example, the number of scenarios is less than the number of processors.

```
pt_shell> remote_execute {set my_paths [get_timing_paths -nworst 10]}
pt_shell> get_distributed_variables my_paths -attributes {slack}
pt_shell> foreach_in_collection path $my_paths(scen1) {
    echo [get_attribute $path slack]
}
pt_shell> remote_execute {set pslack [get_attribute get_pins U1/A] slack}
pt_shell> get_distributed_variables pslack
pt_shell> echo [array get pslack]
{scen1 0.1231 scen2 0.1232}
```

The following example shows how to access an attribute that is a collection.

```
pt_shell> remote_execute {set my_pins [get_pins *]}
pt_shell> get_distributed_variable my_pins -attributes {direction clocks}
pt_shell> foreach_in_collection pin $my_pins(scen1) {
    echo "direction:[get_attribute $pin direction]"
    set my_clocks [get_attribute $pin clocks]
    foreach_in_collection my_clock $my_clocks {
        echo "full clock name : get_attribute $my_clock full_name" }
}
```

In the following example, the slack variable is set at each scenario, and the **get_distributed_variables** command returns the minimum value of the variable.

```
scenario best_case:
set slack 0.3084
```

```
scenario typical_case:
set slack 0.0901
```

```
scenario worst_case:
set slack -0.7081
```

```
pt_shell> get_distributed_variables {slack} -merge_type min
1
pt_shell> echo $slack
-0.7081
```

The following example generates a unique list of all clocks at all scenarios:

```
pt_shell> remote_execute {set clock_names \
    [get_object_name [all_clocks]]}
pt_shell> get_distributed_variables clock_names -merge_type list
1
```

```
pt_shell> echo $clock_names
CLK33 CLK66 CLK50 CLK100 ATPGCLOCK JTAGCLK
```

The following example merges the cells variable from three scenarios with the **unique_list** mode:

```
scenario best_case:
set cells {{U1 U2} {U4 U5} {U7 U8}}
scenario type_case:
set cells {{U2 U3} {U6 U7}}
scenario worst_case:
set cells {{U1} {U5 U6}}
```

```
pt_shell> get_distributed_variables cells -merge_type unique_list
1
pt_shell> echo $cells
{{U1 U2 U3} {U4 U5 U6 U7 U8}}
```

In the following example, the number of scenarios is greater than the number of processors.

```
pt_shell> get_distributed_variables my_paths \
           -pre_commands {set my_paths [get_timing_paths -nworst 10]} -
           attributes slack
```

SEE ALSO

```
set_distributed_parameters(2)
set_distributed_variables(2)
```

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

```
collection get_generated_clocks
[-quiet]
[-regexp]
[-nocase]
[-filter expression]

```

Data Types

<i>expression</i>	string
<i>patterns</i>	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_ilm_objects

Returns a collection of nets, cells, or pins that are part of the interface logic for the current design.

SYNTAX

```
collection get_ilm_objects
[-type {net | pin | cell}]
```

ARGUMENTS

-type {net | pin | cell}

Specifies the type of object to be returned as a collection of objects.

Allowed values are *net*, *pin*, or *cell*. A collection is returned of objects of the specified type that belong to the interface logic on the current design.

DESCRIPTION

Returns a collection of nets, cells, or pins that have been identified as belonging to interface logic by using the **identify_interface_logic** command. The **is_interface_logic_pin** attribute identifies pins that are part of the interface logic. The command takes into account that the design corresponding to the interface logic model (ILM) is flattened, hierarchical nets are reduced to a single flattened net, and hierarchical pins and cells on the original design are not contained in the ILM. The objects returned are those that will be referred to in the interface logic netlist, script, and back-annotation files written using the **write_ilm_*** commands. Use the **get_ilm_objects** command to review the objects that have been identified as belonging to the interface logic on the current design.

For a discussion of ILM creation and the associated commands, see the **identify_interface_logic** man page.

EXAMPLES

The following example returns a collection of all interface logic cells.

```
pt_shell> get_ilm_objects -type cell
```

The following command returns a collection of all nets in the flattened ILM netlist.

```
pt_shell> get_ilm_objects -type net
```

SEE ALSO

```
identify_interface_logic(2)
write_ilm_netlist(2)
write_ilm_parasitics(2)
write_ilm_script(2)
write_ilm_sdf(2)
```

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_group
[-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_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]
objects
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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]
objects
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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.

-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.

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.

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

<i>to_list</i>	list
<i>from_list</i>	list
<i>object_list</i>	list
<i>expression</i>	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  -----" "-----" "-----" \  
          "-----"]  
  
    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]
[patterns | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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

```
int get_license feature_list
```

Data Types

feature_list list

ARGUMENTS

feature_list
Specifies a list of features to be obtained.

DESCRIPTION

Obtains a license for the named features. These features are checked out by the current user until the **remove_license** command is used or until the program is exited. The **get_license** command is valid only when Network Licensing is enabled.

The **list_licenses** command provides a list of the features that you currently have checked out.

EXAMPLES

This example obtains the STAMP Compiler feature:

```
pt_shell> get_license STAMP-Compiler
Checked out license 'STAMP-Compiler'
1
```

SEE ALSO

license_users(2)
list_licenses(2)
remove_license(2)

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

<i>l_id</i>	string
<i>o_id</i>	string
<i>s_id</i>	string
<i>i_id</i>	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_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]
[patterns | -of_objects objects]
```

Data Types

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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:



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 within `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_noiseViolationSources

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

```
int get_noiseViolationSources
[-above]
[-below]
[-low]
[-high]
[-nworst_endpoints pin_count]
[-max_sources_per_endpoint pin_count]
[-slack_type area | height | area_percent]
[object_list]
```

Data Types

<i>pin_count</i>	integer
<i>object_list</i>	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_noiseViolationSources]
```

The following example creates a collection of maximum 10 violation sources for the endpoint ffa/Q above the low rail.

```
pt_shell> get_noiseViolationSources -above -low \
    -max_sources_per_endpoint 10 <HardSpace[get_pin ffa/Q]
```

The following example creates a collection of maximum 10 violation source for the worst two endpoints.

```
pt_shell> get_noiseViolationSources -max_sources_per_endpoint 10 \
    -nworst_endpoints 2
```

SEE ALSO

```
report_noise(2)
report_noiseViolationSources(2)
set_noiseParameters(2)
```

`get_noiseViolationSources`

```
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]

```

Data Types

<i>expression</i>	string
<i>patterns</i>	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_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

<i>expression</i>	string
<i>objects</i>	list
<i>patterns</i>	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

You can use this option only with the *-of_objects* option. For any nets in the *objects* argument to the *-of_objects* option, only pins on leaf cells connected to those nets will be included in the collection. In addition, hierarchical boundaries are crossed in order to find pins on leaf cells.

-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.

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.

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. Note that the *-leaf* option has no effect on the pins of 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_pins
336

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

<i>expression</i>	string
<i>patterns</i>	list
<i>objects</i>	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

<i>expression</i>	string
<i>patterns</i>	list
<i>cells</i>	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

Return 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)`
`report_power_groups(2)`
`remove_power_groups(2)`

get_power_per_pgpin

Reports power on every power pin of the cell

SYNTAX

```
collection get_power_per_pgpin
object_list
```

Data Types

object_list list

ARGUMENTS

object_list
Specifies a list of leaf cells.

DESCRIPTION

The **get_power_per_pgpin** command reports the power on the power pins of the leaf cell. The command output is in the form of a list of lists where every power pin list consists of the power pin name, supply_net or rail name and the power associated with the power pin.

EXAMPLES

In the following example, the **get_power_per_pgpin** command is specified after **update_power**.

```
pt_shell> get_power_per_pgpin U76
{U76 {VDD 'VDD' 4.044e-07} {VNW '' 6.795e-10} {VPW '' 0}}
```

SEE ALSO

```
report_power_pin_info(2)
report_power(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]
[patterns]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	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.

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)
```

get_qtm_ports

```
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 sample_size
  -seed seed
  variation_object Variation object
```

Data Types

<i>sample_size</i>	int
<i>seed</i>	int
<i>variation_object</i>	collection

ARGUMENTS

-sample_size *sample_size*
Specifies the sample size.

-seed *seed*
Specifies the seed for this particular stream. This number must be a positive integer. A different number will generate a different sample.

variation_object
Specifies the variation to generate random samples from. There can only be one object.

DESCRIPTION

Enables the generation of 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 }
_ptsel21
pt_shell> set rnd_list [get_random_numbers -sample_size 500 -
seed 17777 [get_variations test]]
10.613445 14.553537 13.051448 16.382267 ...
```

SEE ALSO

`create_variation(2)`

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

<i>object_type</i>	string
<i>design</i>	string
<i>more</i>	int
<i>fewer</i>	int
<i>num</i>	int
<i>slct_bus</i>	string
<i>target_slct_bus</i>	string
<i>operation</i>	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

Identify the crosstalk bottlenecks in the design. This is useful when the major sources of violations come from crosstalk effects.

SYNTAX

```
int 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

<i>type</i>	list
<i>slack_limit</i>	float
<i>number</i>	int
<i>active_aggressor_count</i>	int

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 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 end points.

```
pt_shell> get_si_bottleneck_nets -cost_type delta_delay -significant_digits 6
```

SEE ALSO

```
report_si_bottleneck(2)
report_default_significant_digits(3)
set_coupling_separation(2)
size_cell(2)
```

get_supply_nets

Create a collection of UPF supply nets in the current design.

SYNTAX

```
string get_supply_nets
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[patterns]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-filter *expression*
Filters the collection with the *expression* option. For any UPF supply nets that match the *patterns* option, 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. 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 supply net names against patterns. Patterns can 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* arguments are mutually exclusive.

DESCRIPTION

The **get_supply_nets** command creates a collection of UPF 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.

For information about collections and the querying of objects, see the **[collections](#)** man page.

EXAMPLES

The following example creates a supply net 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> get_supply_nets {SN1}
{"SN1"}

prompt> get_supply_nets SN*
{"SN1"}
```

SEE ALSO

`create_supply_net(2)`
`report_supply_net(2)`

get_supply_ports

Create a collection of UPF supply ports in the current design.

SYNTAX

```
string get_supply_ports
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[patterns]
```

Data Types

<i>expression</i>	string
<i>patterns</i>	list

ARGUMENTS

-filter *expression*
Filters the collection with the *expression* option. For any UPF 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.

-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 can 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 UPF 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 creates a supply port and creates a collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_port SN1 -domain TOP_DOMAIN
1
pt_shell> get_supply_ports {SN1}
{"SN1"}

prompt> get_supply_ports SN*
{"SN1"}
```

SEE ALSO

`create_supply_port(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

<i>expression</i>	string
<i>patterns</i>	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

```
int 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

<i>state</i>	string
<i>name_list</i>	string
<i>clock_name</i>	string
<i>limit</i>	int
<i>exclusion_group</i>	string
<i>inclusion_group</i>	string
<i>object_list</i>	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 | -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 \
           get_switching_activity
```

```

[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_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

<i>to_list</i>	list
<i>from_list</i>	list
<i>object_list</i>	list
<i>expression</i>	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

Collects timing paths for custom reporting and other processing. You can assign the collection of timing paths to a variable or pass the collection into another command.

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]
[path_collection]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	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
path_collection	collection

ARGUMENTS

- from *from_list*
 Collects paths with startpoints from the specified list of pins, ports, nets, or clocks. Path startpoints are typically the input ports or clock pins of registers. If you specify a clock, all startpoints are considered if they are clocked by the clock.
- rise_from *rise_from_list*
 This option is the same as the **-from** option, except that the path must rise from the specified objects. 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*
 This option is the same as the **-from** option, except that the path must fall from the specified objects. 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*
 Collects paths with endpoints at the specified list of pins, ports, nets, or clocks. Path endpoints are typically the output ports or data pins of registers. If you specify a clock, all endpoints are considered if they are constrained by the clock.
- rise_to *rise_to_list*
 This option is the 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*
 This option is the 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*

Collects paths with throughpoints through the specified list of pins, ports, or nets. Only paths through the specified objects are considered. To specify multiple *through_list* groups, use the **-through** option multiple times. The objects specified within one **-through** option are assumed to be in OR mode. The group of objects specified with multiple **-through** options is assumed to be in AND mode.

- If you specify the **-through** option only one time, the tool collects only the paths that travel through one or more of the objects in the *through_list*.
- If you specify the **-through** option multiple times, the tool collects only the paths that travel through one or more of the objects in each *through_list*. The tool uses the exact order in which the **-through** options are listed; so to obtain correct results, you must ensure that this order is the same as that followed by the actual paths in the circuit.

-rise_through *rise_through_list*

This option is the same as the **-through** option, except that the path must rise through the objects specified.

-fall_through *fall_through_list*

This option is the same as the **-through** option, except that the path must fall through the objects specified.

-exclude *exclude_list*

Excludes all data paths from, through, or to specified list of pins, ports, nets, and cell instances. If a cell instance is specified, all pins of the cell are excluded. The **-exclude** option

- Takes precedence over the **-from**, **-through**, and **-to** options.
- Is exclusive with the **-rise_exclude** and **-fall_exclude** options.
- Does not apply to borrowing path from the **-trace_latch_borrow** option or clock path from the **-path full_clock** or **full_clock_expanded** option.

-rise_exclude *rise_exclude_list*

This option is the same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, or cell instances.

-fall_exclude *fall_exclude_list*

This option is the same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, or cell instances.

-delay_type *delay_type*

Specifies one of the following path delay types, where "rise" or "fall" indicates a rising or falling transition at the path endpoint:

- **max** (the default)

- **min**
 - **min_max**
 - **max_rise**
 - **max_fall**
 - **min_rise**
 - **min_fall**
- nworst paths_per_endpoint**
 Collects the specified number of worst paths per endpoint, where *paths_per_endpoint* is ≥ 1 . The default is 1, meaning that only the worst path to each endpoint is considered. Specifying larger values for *paths_per_endpoint* increases the runtime.
- max_paths max_path_count**
 Collects up to the specified maximum total number of paths among all path groups, where $1 \leq \text{max_path_count} \leq 2000000$; the default is equal to the **-nworst** setting. If you specify *max_path_count* > 1 , and you do not explicitly specify either the **-slack_lesser_than** or the **-slack_greater_than** option, the tool automatically sets **-slack_lesser_than 0** and reports only violating paths.
- group path_group_list**
 Collects only paths that belong to the specified path groups. Paths are created by using the **group_path** or **create_clock** command.
- unique_pins**
 Collects only paths through a unique set of pins. This option can require longer runtimes when used in combination with the **-nworst** option with a large number of paths targeted for collection.
- slack_greater_than minimum_slack**
 Collects only paths with slack greater (more positive) than the *minimum_slack* value. This option is applied as a filter to the paths after they are generated. Therefore, the number of paths generated might be fewer than the number specified with the **-nworst** and **-max_paths** options. To show paths within a specific slack range, use this option together with the **-slack_lesser_than** option.
- slack_lesser_than maximum_slack**
 Collects only paths with slack less (more negative) than the *maximum_slack* value. To show paths within a specific slack range, use this option together with the **-slack_greater_than** option. If you specify *max_path_count* > 1 , and you do not explicitly specify either the **-slack_lesser_than** or the **-slack_greater_than** option, the tool automatically sets **-slack_lesser_than 0** and reports only violating paths.
- ignore_register_feedback feedback_slack_cutoff**
 Ignores timing paths that start and end at the same register that holds a value. Paths are ignored only if the slack is less than the specified *feedback_slack_cutoff* value. This option is applied as a filter to the paths

after they are generated. Therefore, the number of paths generated can be less than the number specified with the **-nworst** and **-max_paths** options. This option applies to minimum delay as well as maximum delay reports.

-include_hierarchical_pins

Collects timing paths that contain points for each hierarchical pin crossed, as well as all leaf pins in the path.

-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 stage, using this option causes the report to retrieve the entire set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. The collection of borrowing paths associated with a given path is stored in the **transparent_latch_paths** attribute of that path. If you use the **-trace_latch_forward** option with this option, the tool appends the trace forward path segments appended to the last path segment.

-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 downstream required then 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 downstream required. In this case, each path segment is reported separately showing the downstream required for that path segment. The **-input_pins**, **-nets**, **-transition_times**, **-capacitance**, and **-significant_digits** options apply to every path segment in the timing report. The remaining options apply only to the first path segment reported. If you use the **-trace_latch_borrow** option is specified with this option, the tool prepends the first path segment with the trace borrow path segments.

-pba_mode none | path | exhaustive

Specifies one of the following path-based analysis modes:

- **none** - Disables path-based analysis.
- **path** - Performs path-based analysis on paths after they have been gathered.
- **exhaustive** - Performs an exhaustive path-based analysis to determine the worst path-based analysis path set in the design. You cannot use the **exhaustive** mode together with the **-start_end_pair** or **path_collection** options.

-normalized_slack

Collects and sorts paths by using normalized slack instead of slack.

Normalized slack divides the slack by an idealized allowed propagation delay. In order to use this option, **update_timing** must have been run with the **timing_enable_normalized_slack** variable set to **true**.

-start_end_pair

Collects paths for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtimes with large memory usage and can lead to generating a huge number of paths depending on the design.

By default, this option searches only for paths that are violating; to change the default behavior, explicitly specify the **-slack_lesser_than** option. You cannot use the **-start_end_pair** option with the **-nworst**, **-max_paths**, **-unique_pins**, **-slack_greater_than**, or **-ignore_register_feedback** option.

Unlike with other options of the **get_timing_paths** command, this option causes the paths returned to no longer be sorted based on slack. Instead, paths are arranged based on the endpoint with those sharing the same endpoint appearing next to one another. The maximum number of paths returned is limited to 2000000. To prevent returning duplicate paths, this option works as though the **timing_report_always_use_valid_start_end_points** variable is set to **true**.

-cover_design

Collects the worst path through each violating pin in the design. A pin is considered to be violating if its slack is less than the value specified by the **-slack_lesser_than** option. You cannot use the **-cover_design** option with the **-nworst**, **-max_paths**, **-unique_pins**, **-ignore_register_feedback**, **-report_ignored_register_feedback**, **-start_end_pair**, or **-pba_mode exhaustive** option. If you use the **-pba_mode path** option, the tool collects paths according to the graph-based analysis (GBA) slack and recalculates. To control the path ordering and slack filtering (GBA or PBA slack), set the **pba_path_mode_sort_by_gba_slack** variable. The maximum number of paths collected is limited to 2000000.

-dont_merge_duplicates

Affects the manner in which paths from multiple scenarios are merged. It turns OFF a main capability in merged reporting that is ON by default. If you do not use this option, 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 into a single instance, but instead shows all critical instances of the path from all scenarios. Since the number of paths reported is limited by the **-nworst**, **-max_paths**, and other options of this command, the resulting merged report might not be evenly spread out across the design, but instead might be focused on the portion of the design that is critical in each scenario. This option is available only if you invoke PrimeTime with the **-multi_scenario** option.

-pre_commands pre_command_string

Specifies a string of commands to execute in the slave context before the execution of merged reporting commands. To group commands in the **pre_command_string**, use 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 execute in the slave context after the execution of merged reporting commands. To group commands in the **post_command_string**, use 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.

-path_type format

Specifies one of the following path report formats and how the timing path is displayed:

- **full_clock_expanded** - Displays the full clock paths between a primary clock and a related generated clock, in addition to the **full_clock** timing path.

-attributes **attribute_list**
Retrieves the specified attributes from a slave collection. This option is available only if you invoke PrimeTime with the **-multi_scenario** option. If you do not use this option, the tool retrieves 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.

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**.

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. The following attributes are supported on timing paths:

clock_uncertainty
endpoint
endpoint_clock
endpoint_clock_close_edge_type
endpoint_clock_close_edge_value
endpoint_clock_is_inverted
endpoint_clock_is_propagated
endpoint_clock_latency
endpoint_clock_open_edge_type
endpoint_clock_open_edge_value
endpoint_clock_pin
endpoint_hold_time_value
endpoint_is_level_sensitive
endpoint_output_delay_value
endpoint_recovery_time_value
endpoint_removal_time_value
endpoint_setup_time_value
object_class
path_group
path_type
points
slack
startpoint
startpoint_clock
startpoint_clock_is_inverted

```
startpoint_clock_is_propagated
startpoint_clock_latency
startpoint_clock_open_edge_type
startpoint_clock_open_edge_value
startpoint_input_delay_value
startpoint_is_level_sensitive
time_borrowed_from_endpoint
time_lent_to_startpoint
```

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. The following attributes are available for points of a timing path:

```
arrival
object
object_class
rise_fall
slack
transition
voltage
depth
distance
applied_derate
derate_factor_depth_distance
incremental
guardband
aocvm_coefficient
```

For information about creating collections and iterating over the elements of a collection, see the man pages of the **collections** and **foreach_in_collection** commands.

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 prints out 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 is 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 paths [get_timing_paths -pba_mode exhaustive
                     -slack_lesser_than 0 -max_paths 1000]
pt_shell> report_timing $paths
```

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_unix_variable

This is a synonym for the **getenv** command.

SEE ALSO

getenv(2), **printenv(2)**, **printvar(2)**, **set(2)**, **setenv(2)**, **sh(2)**, **unset(2)**.

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

<i>group_name</i>	string
<i>weight_value</i>	float
<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>comment_string</i>	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 ($\text{weight} * \text{violation}$), 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. However, for clock gating checks and asynchronous preset/clear checks PrimeTime creates default groups named **"**clock_gating_default**"** and **"**async_default**"** respectively. Any paths belonging to these groups cannot be overridden by a **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_create_category_rule

Creates a category rule for automatic generation of one or more object categories.

SYNTAX

```
string gui_create_category_rule
[-name rule_name]
[-filter filter_spec]
-category category_spec
[-builtin]
```

Data Types

<i>rule_name</i>	string
<i>filter_spec</i>	string
<i>category_spec</i>	string

ARGUMENTS

-name *rule_name*

Specifies the name of the category rule to be created. If this option is omitted, a unique new rule name is automatically generated.

-filter *filter_spec*

Specifies the filter. The basic form of a filter conditional expression is a series of relations joined together with && (and), || (or), and ! (not) operators. Parentheses are used to override precedence. The basic relation is either a Boolean attribute or a comparison of one non-Boolean attribute with another or with a constant value using relational operators.

For example, a filter expression can have any of the following forms:

- **-filter {`endpoint_clock_is_propagated`}** (Boolean attributes)
- **-filter {`(slack < 0)`}** (Comparison with a literal number)
- **-filter {`(path_group.full_name == "inputs")`}** (Comparison with a literal string)

• **-filter {`(startpoint_clock.full_name == endpoint_clock.full_name)`}**
(Comparison of attributes)

Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection. The following relational operators are supported:

- **==** (Equal)
- **!=** (Not equal)
- **==** (Matches pattern)
- **!~** (Does not match pattern)

- &~ (Contains all elements)
- | ~ (Contains some elements)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

The matching operators are used to match wildcard regular expressions. The containment operators are used to compare space-separated set or list attributes.

-category category_spec

Specifies the category. The category specification can be fixed literal text, or it can include an optional object attribute name enclosed in angle brackets; for example, **-category <path_type>**. Additional literal text is optionally allowed to the left and to the right of the angle brackets; for example **-category {Path Type: <path_type>}**. Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection; for example, **-category <startpoint_clock.full_name>**.

When a category rule is "executed" later in the GUI, a separate category (bin of objects) is generated for each unique value of the attribute specified by the **-category** option. (The GUI "category-rule-execution engine" examines attribute values for an input set of objects.)

-builtin

Indicates that the rule being created belongs to the built-in group of category rules. If this option is omitted, the rule being created does not belong to the built-in group.

The built-in group of category rules is a group of category rules that are created before any non-built-in rules. This group of rules includes both built-in rules created by the tool (at tool startup time) and built-in rules that you create before creating any non-built-in rules.

DESCRIPTION

This command creates a category rule. If rule creation is successful, the command returns the name of the newly-created category rule; otherwise, it returns an empty string. After a category rule has been created, it can be used later (at "category-rule execution time" in the GUI) with other rules to generate one or more object categories.

EXAMPLES

This example creates a simple category rule:

```
prompt> gui_create_category_rule -name pathgroups \
      -category <path_group.full_name>
pathgroups
```

This example creates a simple filtering rule:

```
prompt> gui_create_category_rule -name {Negative Slack} \
      -category {Failing Paths} -filter {slack < 0}
Negative Slack
```

This example creates a regular expression filtering rule:

```
prompt> gui_create_category_rule -name {ATPG Sessions} \
      -category {ATPG Sessions} -filter {session_name =~ "atpg*"}
Negative Slack
```

This example creates a set or list containment filtering rule:

```
prompt> gui_create_category_rule -name {Through CPU and CTRL} \
      -category {Through CPU and CTRL} -filter { blocks &~ "CPU CTRL" }
Negative Slack
```

SEE ALSO

```
gui_list_category_rules(2)
gui_remove_category_rules(2)
```

gui_get_category

Returns the contents of the specified category as a collection, including the contents of any offspring categories.

SYNTAX

```
collection gui_get_category
    -category category_hierarchical_name
    [-tree tree_id]
```

Data Types

<i>category_hierarchical_name</i>	list
<i>tree_id</i>	string

ARGUMENTS

-category *category_hierarchical_name*
Specifies the hierarchical name of the category whose contents are returned as a collection. The *category_hierarchical_name* value is a unique hierarchical name of a category in Tool Command Language (Tcl) list format.

-tree *tree_id*
Specifies the string ID of the category tree in which the get-category operation occurs. The current category tree is used if the **-tree** option is not specified.

DESCRIPTION

This command returns the contents of the specified category as a collection, including the contents of any direct or indirect offspring categories.

You should use this command only if at least one category tree is available. The command acts on the specified category tree.

EXAMPLES

The following example gets the contents of the ALL root category as a collection, including the contents of all direct or indirect offspring categories of the ALL root category, and captures the collection returned by the **gui_get_category** command in a Tcl variable:

```
prompt> set category_collection [gui_get_category -category {ALL}]
```

The following example gets the contents of the {Launch Clock: CLK1} category. This category is a child of the {Pathgroup: CLK1} category, which in turn is a child of the ALL root category.

```
prompt> set clct [gui_get_category \
    -category {ALL {Pathgroup: CLK1} {Launch Clock: CLK1}}]
```

SEE ALSO

`gui_create_category_rule(2)`
`gui_list_category_rules(2)`
`gui_remove_category_rules(2)`

gui_get_cell_block_marks

Gets a list of the block mark string values on one or more cells.

SYNTAX

```
list gui_get_cell_block_marks
cells
```

Data Types

cells list

ARGUMENTS

cells
Lists one or more cell name patterns or cell collections.

DESCRIPTION

This command gets the block mark string values from one or more hierarchical or leaf-level cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU
prompt> gui_get_cell_block_marks [get_cells I_TOP/I_ALU]
ALU
```

The following example shows how using the **get_attribute** command to query the **block_mark** attribute is an alternative to using the **gui_get_cell_block_marks** command:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
prompt> get_attribute -class cell I_TOP/I_ALU block_mark
ALU
```

SEE ALSO

`gui_set_cell_block_marks(2)`
`gui_remove_cell_block_marks(2)`

gui_list_category_rules

Lists all category rules except built-in rules by default.

SYNTAX

```
list gui_list_category_rules
[-all | -names rule_names_list]
[-format script | tcl_list]
```

Data Types

rule_names_list list

ARGUMENTS

-all

Lists all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-names *rule_names_list*

Specifies the names of the category rules to be listed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-format *script* | *tcl_list*

Specifies the output format in which the category rules are listed. The default output format is **script**.

When the **script** format is used, the category rules are listed as a Tool Command Language (Tcl) script of **gui_create_category_rule** commands that you can replay. In this case, the rules are streamed to the output stream. You can redirect the output by using the **redirect** command, for example, if you want to capture the script output in a file to use again later. When the **script** format is used, the command result is an empty string.

If the **tcl_list** format is specified, the category rules are listed in an "array set compatible" Tcl list format. In this case, the command result is a Tcl list.

DESCRIPTION

This command lists category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you run this command without specifying either the **-all** option or the **-names** option, the command lists all category rules except built-in rules. Specify the **-all** option to list all category rules, including the built-in rules. Specify the **-names** option to list individual rules by name.

EXAMPLES

The following example lists all category rules, including the built-in rules:

```
prompt> gui_list_category_rules -all
gui_create_category_rule -name Pathgroups -builtin \
    -category <path_group.full_name>
gui_create_category_rule -name Session -builtin \
    -category <session_name>
gui_create_category_rule -name {Path Type} -builtin \
    -category <path_type>
...
...
```

The following example lists all category rules except the built-in rules:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> gui_list_category_rules
gui_create_category_rule -name rule1 \
    -category <session_name>
gui_create_category_rule -name rule2 \
    -category <path_group.full_name>
```

The following example redirects the script output to a file that you can source during a subsequent run of the tool:

```
prompt> redirect -file /remote/dir1/rules.tcl {gui_list_category_rules}
prompt>
```

The following example uses the **-names** option to list only the rules named rule1 and rule3:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> gui_create_category_rule -name rule3 -category <startpoint.full_name>
rule3
prompt> gui_list_category_rules -names {rule1 rule3}
gui_create_category_rule -name rule1 \
    -category <session_name>
gui_create_category_rule -name rule3 \
    -category <startpoint.full_name>
```

The following example uses the **tcl_list** output format to query the category specification of the rule named rule1:

```
prompt> gui_create_category_rule -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule -name rule2 -category <path_group.full_name>
rule2
prompt> array set rules [gui_list_category_rules -format tcl_list]
```

```
Information: Defining new variable 'rules'. (CMD-041)
prompt> array set rule1_options $rules(rule1)
Information: Defining new variable 'rule1_options'. (CMD-041)
prompt> set rule1_cat_spec $rule1_options(category)
Information: Defining new variable 'rule1_cat_spec'. (CMD-041)
<session_name>
```

SEE ALSO

`gui_create_category_rule(2)`
`gui_remove_category_rules(2)`

gui_list_cell_block_marks

Lists the cell names and block mark values for cells that are marked in the design.

SYNTAX

```
list gui_list_cell_block_marks
```

ARGUMENTS

This command has no arguments

DESCRIPTION

This command lists the cell names and block mark values for all cells that are marked in the design. The listing is sorted by the full names of the marked cells.

EXAMPLES

The following example sets block marks on two hierarchical cell instances, and then lists the names of all the marked cells and their block marks:

```
prompt> gui_remove_cell_block_marks -all
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_set_cell_block_marks I_TOP/I_REG_FILE  REG_FILE
prompt> gui_list_cell_block_marks
I_TOP/I_ALU      ALU
I_TOP/I_REG_FILE   REG_FILE
```

SEE ALSO

```
gui_set_cell_block_marks(2)
gui_get_cell_block_marks(2)
gui_remove_cell_block_marks(2)
```

gui_remove_category_rules

Removes all category rules except built-in rules by default.

SYNTAX

```
string gui_remove_category_rules
[-all | -names rule_names_list]
```

Data Types

rule_names_list list

ARGUMENTS

-all

Removes all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

-names rule_names_list

Specifies the names of the category rules to be removed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

DESCRIPTION

This command removes category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you use this command without specifying any options, it removes all category rules except built-in rules. Specify the **-all** option to remove all category rules, including the built-in rules. Specify the **-names** option to remove individual rules by name.

The command returns an empty string as its result.

EXAMPLES

The following example removes all category rules, including the built-in rules:

```
prompt> gui_remove_category_rules -all
```

The following example removes all category rules except the built-in rules:

```
prompt> gui_remove_category_rules
```

The following example removes the rules named Rule1 and Rule2:

```
prompt> gui_remove_category_rules -names {Rule1 Rule2}
```

The following example removes non-built-in rules at the top of a user-defined category rule creation script, which is being developed interactively. You can repeatedly change and refine this script by using a text editor, and you can source the script multiple times in a single run of the tool.

```
# first remove all existing user-defined non-built-in rules  
gui_remove_category_rules  
# now create the user-defined category rules  
gui_create_category_rule ...  
gui_create_category_rule ...  
and so forth
```

SEE ALSO

[gui_create_category_rule\(2\)](#)
[gui_list_category_rules\(2\)](#)

gui_remove_cell_block_marks

Removes the block mark string values from one or more cells.

SYNTAX

```
status gui_remove_cell_block_marks
-all | cells
```

Data Types

cells list

ARGUMENTS

-all

Removes all block marks on all cells that are marked.

This option and the *cells* option are mutually exclusive, and one of them must be specified.

cells

Removes block marks from one or more cells specified in a list of cell name patterns or cell collections.

This option and the **-all** option are mutually exclusive, and one of them must be specified.

DESCRIPTION

This command removes the block mark string values from one or more hierarchical or leaf-level cell instances. If the **-all** option is specified, then all the block marks are removed from all the cell instances that are marked. If the *cells* option is specified, then block marks are removed only from the specified cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
prompt> gui_remove_cell_block_marks I_TOP/I_ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU
prompt> gui_remove_cell_block_marks [get_cells I_TOP/I_ALU]
```

The following example removes all the block marks from all the cell instances that are marked:

```
prompt> gui_remove_cell_block_marks -all
```

SEE ALSO

`gui_set_cell_block_marks(2)`
`gui_get_cell_block_marks(2)`
`gui_list_cell_block_marks(2)`

gui_set_cell_block_marks

Sets a block mark on one or more cells.

SYNTAX

```
status gui_set_cell_block_marks
cells
block_mark
```

Data Types

<i>cells</i>	list
<i>block_mark</i>	string

ARGUMENTS

cells

Lists one or more cells to mark with a block mark, in a list of cell name patterns or cell collections.

block_mark

Specifies the block mark string value to be set for the listed cells.

DESCRIPTION

This command sets a block mark for one or more hierarchical or leaf-level cell instances. Typically, a short symbolic string name is used as the block mark. Ultimately, a block mark can appear in the GUI as the text label for a timing path category generated from a dynamic category rule. Such block marks should be kept short to avoid using up too much screen space in the GUI.

A common application involves marking certain interesting intellectual property (IP) blocks within a design. These block marks affect the calculation of the following timing path shell attributes, which are defined only when the GUI is running:

- **blocks** - An ordered block level path; includes start, through, and end blocks
- **through_blocks** - Includes through blocks, but not the start and end blocks
- **start_block** - The start block
- **end_block** - The end block
- **start_end_blocks** - A pair consisting of the start block and the end block, in that order

- **start_end_blocks_sorted** - A pair consisting of the start block and the end block, sorted alphabetically

These block marks also affect the calculation of the following cell shell attribute, which is defined only when the GUI is running:

- **block_mark** - String value of a cell instance's (explicit) block mark, or an empty string if there is no explicit block mark

In calculating the timing path shell attributes listed previously, a block is calculated for each timing point of a timing path. If the leaf cell instance on which a timing point resides is explicitly marked with a block mark, by using the **gui_set_cell_block_marks** command, the block for the timing point is the block mark for the leaf cell. If the leaf cell instance on which a timing point resides is not explicitly marked, the block for the timing point is the block mark on the first explicitly marked hierarchical cell located by traversing up the logical hierarchy from that leaf cell instance.

If no direct or indirect hierarchical parent cell is found to be explicitly marked, the block for the timing point has a default implicit string value of "_Top_". For a timing point that is a startpoint or endpoint of a timing path, where the startpoint or endpoint is an external design port, the block has a default implicit string value of "_Port_".

Note: Each timing point of a timing path always has a defined block, which can be an explicit block mark, "_Top_", or "_Port_".

The **through_blocks** attribute value can be an empty set of blocks, which is represented by the string value "_Empty_".

EXAMPLES

The following example sets the block mark to ALU for a hierarchical cell instance identified by a cell name:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU ALU
```

The following example sets the block mark to ALU for a hierarchical cell instance identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU] ALU
```

The following example sets the block mark for a list of cell instances where the first item is identified by a cell name and the second item is identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [list I_ALU [get_cells I_STACK_TOP/
I1_STACK_MEM]] \
my_mark
```

SEE ALSO

`gui_get_cell_block_marks(2)`
`gui_remove_cell_block_marks(2)`

gui_start

Starts the PrimeTime GUI.

SYNTAX

```
string gui_start
      [-file name_of_script_file]
      [-no_windows]
```

Data Types

name_of_script_file string

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 PrimeTime GUI from the pt_shell prompt. It is ignored if the PrimeTime GUI has already been started.

EXAMPLES

The following example starts the PrimeTime GUI.

```
pt_shell> gui_start
```

SEE ALSO

`gui_stop(2)`

gui_stop

Stops the PrimeTime graphical user interface (GUI).

SYNTAX

string **gui_stop**

ARGUMENTS

None.

DESCRIPTION

This command stops the PrimeTime GUI and returns to the pt_shell prompt. It is ignored if the PrimeTime GUI has not been started or has previously been stopped.

EXAMPLES

The following example stops the PrimeTime GUI and returns to the pt_shell prompt.

```
pt_shell> gui_stop
```

SEE ALSO

[gui_start\(2\)](#)

gui_write_window_image

Write a specified window to an image file.

SYNTAX

```
status gui_write_window_image
-file file_name
[ -format png | xpm | jpg | bmp ]
[ -window window_id ]
```

Data Types

file_name string
window_id string

ARGUMENTS

```
-file file_name
      Specifies the name of the file where the image will be saved.
      The file name's postfix will determine the image format, unless -format has
      been used.

-format png | xpm | jpg | bmp
      Specifies the image format to be written to the file.
      Default value is png.
      Note: If the format value contradicts the file_name postfix the format will
      take precedence and be appended to file_name.

-window window_id
      Specifies the window name to write to the image file.
      Open window ids can be queried using gui_get_window_ids command.
      Default value is most recently used window.
```

DESCRIPTION

The **gui_write_window_image** command writes to a specified file an image of a window.

EXAMPLES

The following example will create a toplevel window and a child schematic view window, then will write out the schematic window's image to specified file *my_image.xpm*.

```
shell> set top [gui_create_window -type TopLevel]
shell> set schm [gui_create_schematic]
shell> gui_write_window_image -file my_image.xpm -window $schm
```

SEE ALSO

gui_create_schematic(2)

```
gui_create_window(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)
```

gui_write_window_image
410

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)`

help_attributes

Display help for attributes and object types

SYNTAX

```
string help_attributes [-verbose] [-application] [-user] [class_name] [attr_pattern]  
string class_name  
string attr_pattern
```

ARGUMENTS

```
-verbose  
    Display attribute properties.  
  
-application  
    Only display application defined attributes.  
  
-user  
    Only display user defined attributes.  
  
class_name  
    Object class to retrieve help for.  
  
attr_pattern  
    Show attributes matching pattern.
```

DESCRIPTION

The **help_attributes** command is used to get quick help for the set of available attributes. When this command is run with no arguments a brief informational message is printed followed by the available object classes.

EXAMPLES

```
prompt> help  
cc1_test> help_attributes  
Specify an object type and an optional wild card pattern to get information  
on the available attributes defined for the specified object type. Use the  
-verbose option to get detailed information on the attributes
```

The available object types are:

```
cell          net          pin  
...  
...
```

SEE ALSO

```
get_attribute (2)  
help (2)  
get_defined_attributes (2)
```

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

history

416

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

```
int identify_interface_logic
[-ignore_ports port_list]
[-auto_ignore]
[-latch_level levels]
[-context_borrow]
[-keep_ignored_fanout]
[-include_pins pins_list]
[-critical_pins]
[-include_all_net_pins]
[-traverse_disabled_arcs]
```

Data Types

port_list	string
level	int

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 impacting 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 pins_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 impacted by, or impacts, 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 -level 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 IO 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)
```

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

<i>collection1</i>	collection
<i>index</i>	int

ARGUMENTS

<i>collection1</i>	Specifies the collection to be searched.
<i>index</i>	Specifies the index into the collection. Allowed values are integers 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 example from PrimeTime uses the **index_collection** command to extract the first object of a collection.

```
pt_shell> set c1 [get_cells {u1 u2}]\n{ \"u1\", \"u2\"}\npt_shell> query_objects [index_collection $c1 0]\n{ \"u1\"}
```

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]
```

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 includes 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
```

Objects	Type	Current Static Probability	Current Toggle Rate	Proposed Static Probability	Proposed Toggle 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
```

Driver	Current Static Probability	Current Toggle Rate	Proposed Static Probability	Proposed Toggle 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

<i>lib_spec</i>	list
<i>new_net_names</i>	list
<i>new_cell_names</i>	list
<i>pin_or_port_list</i>	list
<i>lib_cell</i>	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 i.e. 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 i.e. 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:

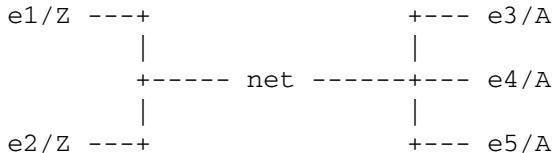
```
e1/Z --- old_net --- e3/A
```

creates the following:

```
e1/Z --- old_net --- U1 --- net1 --- e3/A
```

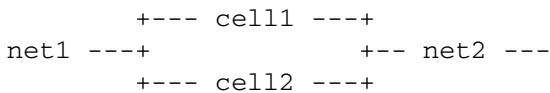
- If the pin is a load, it is disconnected from its net, and connected to the new net.
- If the pin is a driver, all loads on that net are disconnected from the old net and connected to the new net.
- When pins on the same net are specified, they are grouped, and one buffer is inserted. Loads and drivers are grouped separately.

- If the net is multi-driven, 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.



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.



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.

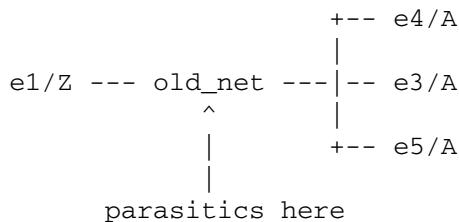
Buffering and its Effects on Parasitics

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)

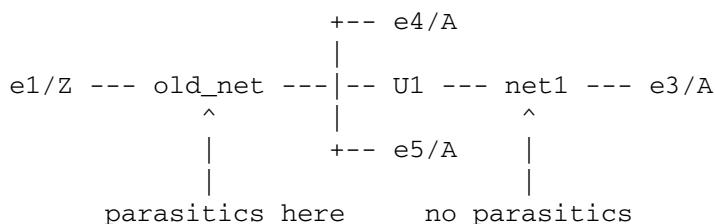
Buffering load pins in the presence of parasitics.

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:

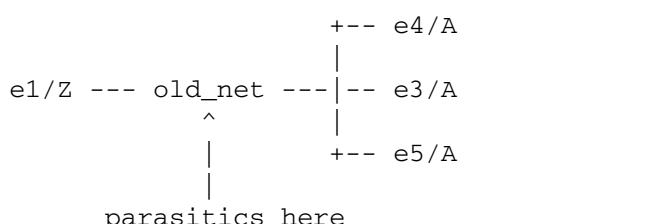


results in the parasitics being kept on the old net

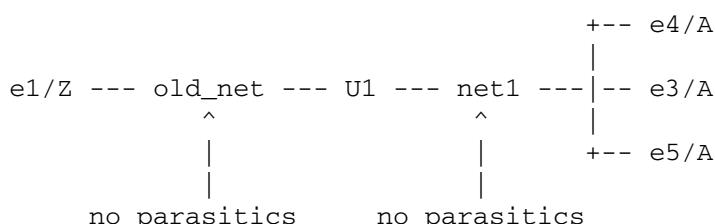


Buffering more than a single load pin results in the parasitics on the net connected to the input of the inserted buffer being removed and a warning being issued. (See PARA-060)

Buffer load pins e3/A, e4/A and e5/A



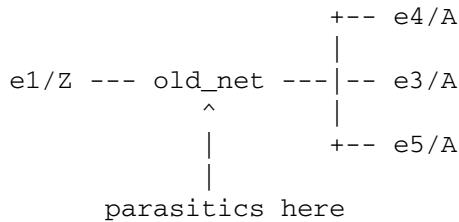
results in the parasitics being removed



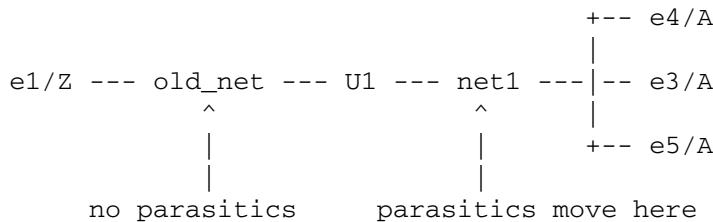
Buffering driver pins in the presence of parasitics.

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

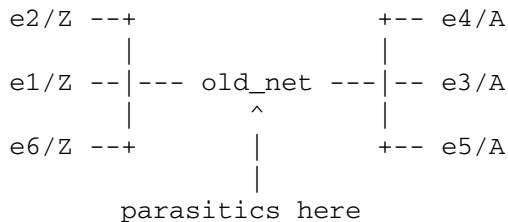


results in the parasitics being moved to net1

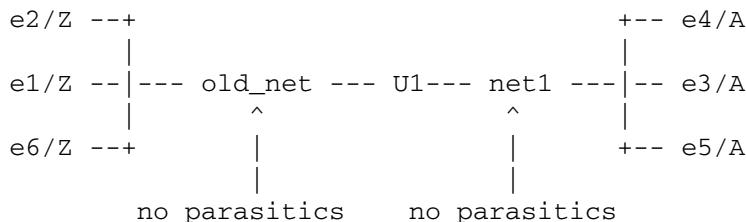


Buffering more than a single driver pin on a multiply driven net results in the parasitics being removed from the net connected to the output of the inserted buffer. (See PARA-060)

Buffer driver pin e1/Z, e2/Z and e6/Z as follows:



results in the parasitics being removed



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.

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

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

is_false
```

```
? }
```

```
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] } {
```

is_true

438

```
?           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

```
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

Displays information about all licenses, related users, and host names currently in use. If a feature list is specified, only information about those features is displayed.

The **license_users** command is valid 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

The **link** command, a synonym for the **link_design** command, exists in PrimeTime for compatibility with Design Compiler.

SEE ALSO

[link_design\(2\)](#)

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

Indicates that the linker displays verbose messages.

-force

By default, if the target design is already fully linked, it is not relinked. This option forces relinking of the design.

-remove_sub_designs

Indicates that subdesigns are removed after linking. By default, subdesigns are removed. Use this option to free up memory and improve performance. For more information, see the "Performance Considerations" section.

-keep_sub_designs

Indicates that subdesigns are kept 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 name of the design to be linked; the default is the current design.

DESCRIPTION

Performs a name-based resolution of design references for the specified *design_name* variable or the current design. In addition to resolving references in the design, the **link_design** command builds the internal representation of the design for analysis.

A complete, fully functional design must be connected to all referenced library components and designs; the references must be located and linked to the current design. Thus, the purpose of this command is to locate all designs and library components referenced by the current design and link them to the current design. During the link, all files specified in the **link_path** command are loaded if they are

not already in memory. The goal is a fully instantiated design on which analysis can be performed.

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 this behavior setting the **link_force_case** variable.

Automatic Loading of Designs and Libraries

You can set the **link_path** and **search_path** variables so that you need to read only your top design and then link. PrimeTime automatically finds and loads other designs and libraries that are needed.

In the following example, assume that your main design is in newcpu.db and uses the cmos.db library. The **link_path** variable specifies cmos.db, so the linker loads cmos.db when it starts. As the link proceeds, if a design is required and is not in memory, the linker searches the **search_path** variable for a file named *reference_name*.db. For example, the referenced BOX1 design is not in memory, so the linker 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 linker fails to resolve one or more references, first determine and correct

the source of the problem, then relink the design using the **link_design** command. Failures are typically caused by missing libraries or designs; incorrectly specified **link_path** or **search_path** variable; or a file that is in the path but is not accessible by you. After making the necessary corrections to your environment, you can decide whether to do an incremental relink or an initial link, or whether to allow the linker 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). You can prevent unresolved references by ensuring that the **link_create_black_boxes** variable is set 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.

Link With Mismatches

By default, if there are pin mismatches between instance and reference, linker issues errors and fails. When **link_allow_design_mismatch** is **true**, linker issues warnings and still succeeds. This allows users to gather useful information even when part of a design is in early stage.

Performance Considerations

The tool starts linking with the top design along with any other designs and libraries you loaded. 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 maintain the subdesigns if you want to

- Keep all designs and analyze them later

- Relink the design later with the **-force** option

Note that this 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)`

```
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
<hr/>				
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_ss	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
<hr/>				
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)`

`list_attributes`

```
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. These include 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. By default, only designs that are currently in memory are listed. When used with the **-all** option, the **list_designs** command also lists the designs that are instantiated in a linked design but have been removed from memory.

You can filter unused designs out of the display by using the **-only_used** option. Designs that are in use include the current design, any design that is linked or partially linked, and any designs instantiated in a linked design.

The **list_designs** command displays the status of the design. The notation used is

- * - Indicates that the design is the current design
- L - Indicates that the design is linked.
- N - Indicates that the design is not in memory.
- l - Indicates that the design is partially linked.

EXAMPLES

The following command lists the designs in memory.

```
pt_shell> list_designs
```

Design Registry:

*L AD4FULA	/u/foo/designs/top.db:AD4FULA
AD4PG	/u/foo/designs/add1.db:AD4PG
ADD5A	/u/foo/designs/add2.db:ADD5A
MULT1	/u/foo/designs/top.db:MULT1

list_designs

The following command lists only the designs that are in use and in memory.

```
pt_shell> list_designs -only_used
Design Registry:
*L AD4FULA      /u/foo/designs/top.db:AD4FULA
    ADD5A        /u/foo/designs/add2.db:ADD5A
    MULT1        /u/foo/designs/top.db:MULT1
```

After removing design MULT1, 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      /u/foo/designs/top.db:AD4FULA
    ADD5A        /u/foo/designs/add2.db:ADD5A
    N MULT1      /u/foo/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

```
int 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 all libraries that are read into PrimeTime.

SYNTAX

```
string list_libs
[-only_used]
```

ARGUMENTS

-only_used

Indicates only the list 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.

An asterisk (*) to the left of a library indicates that this is the main library for the current design. The main library is the first library in the link path which has a time unit. An uppercase 'M' to the left of a library indicates that the library is the maximum library of a max/min library relationship created by the **set_min_library** command. A lowercase 'm' to the left of a library indicates that the library is the minimum library of a max/min library relationship created by the **set_min_library** command.

This command was named as **list_libraries** prior to 2011.06 release.

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 lib 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

```
string list_licenses
```

DESCRIPTION

The **list_licenses** command lists the licenses which you currently have checked out. The **list_licenses** command always returns the empty string.

EXAMPLES

This example shows the output from the **list_licenses** command.

```
pt_shell> list_licenses
```

Licenses in use:

```
    PrimeTime  
    PrimeTime-SI
```

SEE ALSO

```
get_license(2)  
license_users(2)  
remove_license(2)
```

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 11 {a b c}
a b c

prompt> set 12 [lminus $11 {a b d}]
c
```

```
prompt> set 13 [lminus $11 d]
a b c
```

The following example illustrates the use of **lminus** with the **-exact** option:

```
prompt> set 11 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c
```

```
prompt> set 12 [lminus $11 a*]
{b[1]} b c
```

```
prompt> set 13 [lminus -exact $11 a*]
a {a[1]} {b[1]} b c
```

```
prompt> set 14 [lminus -exact $11 {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_upf

Reads in a script in the Unified Power Format (UPF) format.

SYNTAX

```
status load_upf
[-scope instance_name]
[-version upf_version]
[-supplemental supp_file_name]
file_name
```

Data Types

<i>instance_name</i>	string
<i>upf_version</i>	string
<i>supp_file_name</i>	string
<i>file_name</i>	string

ARGUMENTS

-scope *instance_name*
Specifies the scope where the UPF commands contained in the *file_name* file are to be executed. If you do not specify the **-scope** option, the tool uses the current scope.

-version *upf_version*
Specifies the version of UPF to which the file conforms. Allowed version is 1.1 (the default).

-supplemental *supp_file_name*
Specifies the supplemental UPF file name to be read. This option is only supported in Golden UPF mode.

file_name
Specifies the name of the file that contains the UPF script to be read. If you are not using the golden UPF mode, this argument is required. In golden UPF mode, this argument is optional and specifies the name of the golden UPF file.

DESCRIPTION

The **load_upf** command sets the scope to the specified instance and executes the set of UPF commands in the *file_name* file. Upon return, the current scope is restored to what it was before invocation. The commands are executed the same way as the **source** Tcl command.

EXAMPLES

The following example loads the top.upf script.

```
prompt> load_upf top.upf
1
```

SEE ALSO

`create_power_domain(2)`
`source(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

```
string map_design_mode
design_mode
[-from from_pin_list]
[-to   to_pin_list]
[-through through_pin_list]
[cell_mode_list]
[instance_list]
```

Data Types

<i>design_mode</i>	list
<i>from_pin_list</i>	list
<i>to_pin_list</i>	list
<i>through_pin_list</i>	list
<i>instance_list</i>	list
<i>cell_mode_list</i>	list

ARGUMENTS

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.

-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.

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.

A design mode is mapped to a cell mode using the **map_design_mode cell_mode_list instance_list design_mode** command. If the design mode changes from inactive to active, then all cell modes mapped to the design mode also become active.

A path is mapped to a design mode using the **map_design_mode -from [from_pin_list] -to [to_pin_list] -through [through_pin_list] design_mode** command. 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

```
map_design_mode(2)
define_design_mode_group(2)
report_mode(2)
set_mode(2)
reset_mode(2)
```

mem

Shows the total memory used by PrimeTime.

SYNTAX

integer **mem**

ARGUMENTS

This command has no arguments.

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_models

Merges multiple timing models (in LIB format) together to be one.

SYNTAX

```
string merge_models
[-lib_files lib_file_names]
-mode_names mode_names
[-group_name mode_group_name]
-output file_name
[-formats format_list]
[-tolerance merge_tolerance]
[-pin_cap_tolerance merge_tol_of_pin_cap]
[-single_mode]
[-keep_all_arcs]
```

Data Types

<i>lib_file_names</i>	list
<i>mode_names</i>	list
<i>mode_group_name</i>	string
<i>file_name</i>	string
<i>format_list</i>	list
<i>merge_tolerance</i>	float
<i>merge_tol_of_pin_cap</i>	float

ARGUMENTS

-lib_files *lib_file_names*
Specifies a list of Synopsys internal library (.lib) files to read, compile, and merge. The files must have been extracted using the **extract_model** command. Substitute the list you want for *lib_file_names*.

-mode_names *mode_names*
Specifies a list of mode names for the list of models. The mode names must match the list of library files specified. Substitute the list you want for *mode_names*.

-group_name *mode_group_name*
Specifies the name of the mode group to which the models belong. If you do not specify this option, the default group name is *etm_modes*.

-output *file_name*
Specifies the name of the output file that is used to save the merged model. Substitute the name you want for *file_name*.

-formats *format_list*
Specifies the formats the merge model is written in, either db, lib, or both. Substitute the formats that you want for *format_list*.

-tolerance *merge_tolerance*
Specifies the float tolerance used to compare delay or transition values of

arcs in the timing models. It defines the tolerance for floating point number comparison when the timing arcs in the models are matched and compared against each other to determine they are equal. Float numbers are considered equal if the difference between them is within the tolerance.

The default is 0.04. The values from the first .lib file listed in the **-lib_file** option is used in the merged model. Substitute the tolerance you want for *merge_tolerance*; however, the value must be greater than or equal to zero.

This option cannot be used with the *keep_all_arcs* option.

-pin_cap_tolerance merge_tol_of_pin_cap

Specifies the float tolerance used to compare the capacitance attribute values for the matching pins across the models. Float numbers are considered equal if the absolute difference between them is within the specified tolerance.

The default is 0.002. The values from the first .lib file listed in the **-lib_file** option is used in the merged model if equal within tolerance.

-single_mode

Forces one timing arc to have only one defined mode or forces there to be only one single mode per timing arc in the merged model. This is necessary for downstream tools that cannot handle more than one mode on a timing arc. This normally results in larger, less compact timing models.

This option cannot be used with the *keep_all_arcs* option.

-keep_all_arcs

This makes the *merge_models* option ignore the *merge_tolerance* and suppresses all arc merging, and it implies *single_mode*. Use this option when performing two independent merges and the resulting merged models must have the same number of arcs. Resulting models are very large. This option cannot be used with either the *single_model* or *merge_tolerance* option.

DESCRIPTION

Merges multiple timing models together to be one. The timing models must be in Synopsys internal library (.lib) format. This command reads in all the specified files to generate a list of timing models. These timing models are processed, matched, compared, and merged to create a single timing model. This model has moded timing arcs such that, in any given mode, the static timing behavior of the merge model is equivalent to one of the models merged. The final merged model is saved on disk in the specified files and requested formats. PrimeTime and Design Compiler recognize the merged model with moded timing arcs for performing timing analysis.

The *keep_all_arcs* argument should only be used when a timing model has been extracted under several different conditions and the results are to be merged into two or more models in situations where the number of arcs must match. For example, say that a model TEST is extracted with two modes A and B, operating conditions min and max. There are four extractions, min A, min B, max A and max B. Now you want to merge all min values for modes A and B to get a minimum model, and all max values to get a maximum model. If you were going to use the merged models as the minimum and maximum libraries of the **set_min_library** command, their arcs must match. Therefore, you should merge the models using the *keep_all_arcs* option.

EXAMPLES

The following command reads in the LIB models specified by the sram_write.lib and sram_read.lib files and generates a merged model containing an "SRAM_MODE" mode group with two "READ" and "WRITE" component modes. Timing arcs existing only in the sram_write.lib file are put into the merged model and marked with "WRITE" mode. Timing arcs existing only in the sram_read.lib file are marked with "READ" mode. Timing arcs existing in both models are compared to each other by their delay/transition values. If they are within 0.1 time units of each other, they are considered equal, and only one is kept in the merged model with no mode marked on the arc (valid for both modes); otherwise both arcs are kept in the merged model with each arc marked with a proper mode depending on which model it came from. Because the command requests all formats, the merged model is saved as sram_model.lib.db in Synopsys database format (.db), and sram_model.lib in Synopsys internal library format (.lib).

```
pt_shell> merge_models -lib_files \
           {sram_write.lib sram_read.lib} \
           -mode_names {WRITE READ} -mode_group SRAM_MODE \
           -output sram_model -format {db lib} \
           -tolerance 0.1
```

SEE ALSO

```
extract_model(2)
set_min_library(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]
  [-output merged_saif_name]
  [-simple_merge]
  [-ignore ignore_name]
  [-exclude exclude_file_name]
  [-derate_glitch value]
  [-quiet]
```

Data Types

<i>saif_file_and_weight_list</i>	list
<i>inst_name</i>	string
<i>prefix</i>	string
<i>merged_saif_name</i>	string
<i>ignore_name</i>	string
<i>exclude_file_name</i>	string
<i>value</i>	float

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.

-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.

-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.

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> merge_saif -input_list { -input gate_back1.saif -weight 10 \
           -input gate_back2.saif -weight 20 \
           -input gate_back3.saif -weight 30 \
```

```
-input gate_back4.saif -weight 40 } \  
-strip_path E/UUT -output merged_saif.saif
```

SEE ALSO

`set_switching_activity(2)`
`set_rtl_to_gate_name(2)`
`get_switching_activity(2)`
`reset_switching_activity(2)`
`report_switching_activity(2)`
`write_saif(2)`
`report_power(2)`

parallel_execute

Specifies a list of commands to be executed in parallel and their output files.

SYNTAX

```
status parallel_execute
[-commands_only]
[-compress]
concurrent_cmds
```

Data Types

concurrent_cmds list

ARGUMENTS

```
-commands_only
    Prints output of commands after all commands complete execution.

-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 -commands_only option.

concurrent_cmds
    The list of commands, if the option -commands_only is specified. Otherwise,
    it is a list of interleaved strings that are commands and their output files.
    When there are sufficient computing resources available, these commands are
    to be executed in parallel with their results written into the specified
    output files separately.
```

DESCRIPTION

This command is a parallel directive that instructs PrimeTime to execute the list of commands in parallel. The output of commands is either printed out (if the option **-commands_only** is specified) or written out to the specified log files. It enables you to exploit parallelism across unrelated PrimeTime Tcl commands.

The following syntax shows how to use the **parallel_execute** command:

```
update_timing -full
parallel_execute {
    report_cmd1 rpt_file1
    report_cmd2 rpt_file2
    ...
    report_cmdN rpt_fileN
}
```

Any post-update reporting and analysis commands are applicable.

The **parallel_execute** command honors the resource constraint specified by the **set_host_options** command with the **-local_process** and **-max_core** options.

When there are sufficient resources, executing multiple PrimeTime post update reporting and analysis commands in parallel can reduce the overall runtime of the script, comparing to when those reporting commands are executed sequentially. The commands recommended for the **parallel_execute** command include many common post-update commands, such as the **report_clock_timing**, **check_timing**, **report_bottleneck**, **report_exceptions**, **report_min_pulse_width**, **save_session**, path-based analysis, **create_ilm**, and **extract_model** commands or even user-defined Tcl procedures that are custom reports based on these native PrimeTime reporting commands.

To get the best results, it is important to understand that the **parallel_execute** command should contain post update reporting and analysis commands. Any commands that cause changes to the computed timing data and trigger incremental update result in severe errors and script and command execution immediately stops.

The **parallel_execute** command honors the resource constraint specified by the **set_host_options** command with the **-local_process** and **-max_core** options.

Note that the number of commands allowed inside the **parallel_execute** command has no practical limit, and PrimeTime automatically and dynamically schedules these commands if there are more commands than number of computing resources (CPUs or cores) available.

Also note that the **parallel_execute** command is itself a blocking command, which means PrimeTime does not execute the next command following the **parallel_execute** command after finishing all the commands specified in the list. To get closest to optimal speed-up, it is recommended that the longer runtime commands be specified first.

EXAMPLES

In the following example, four cores are allocated on the local host to perform parallel execution. Next, after the **update_timing** command, the **report_timing**, **report_constraints**, **save_session** PrimeTime commands and user Tcl procedure **report_qor** are executed in parallel.

```
pt_shell> set_host_options -local -max_cores 4
...
pt_shell> update_timing
pt_shell> parallel_execute {
    {report_timing -max 10000 -nworst 10} rpt_tim.log
    {report_constraints -all_violators} rpt_cstr.log
    {report_qor} rpt_qor.log
    {save_session design_dir/my_session} /dev/null
}
```

In the second example, **parallel_execute** is used to evaluate and print out the output of commands **check_timing** and **report_analysis_coverage**.

```
pt_shell> parallel_execute -commands_only {
    {check_timing}}
```

```

        {report_analysis_coverage}
    }
Output of command 'check_timing':
Information: Checking 'no_input_delay'.
Information: Checking 'no_driving_cell'.
Information: Checking 'unconstrained_endpoints'.
Information: Checking 'unexpandable_clocks'.
Information: Checking 'latch_fanout'.
Information: Checking 'no_clock'.
Information: Checking 'partial_input_delay'.
Information: Checking 'generic'.
Information: Checking 'loops'.
Information: Checking 'generated_clocks'.
Information: Checking 'pulse_clock_non_pulse_clock_merge'.
Information: Checking 'pll_configuration'.
check_timing succeeded.

```

Output of command 'report_analysis_coverage':

```
*****
Report : analysis_coverage
Design : counter
...
*****
```

Type of Check	Total	Met	Violated	Untested
setup	5	5 (100%)	0 (0%)	0 (0%)
hold	5	4 (80%)	1 (20%)	0 (0%)
out_setup	5	5 (100%)	0 (0%)	0 (0%)
out_hold	5	5 (100%)	0 (0%)	0 (0%)
All Checks	20	19 (95%)	1 (5%)	0 (0%)

SEE ALSO

```
redirect(2)
set_host_options(2)
```

parallel_FOREACH_IN_COLLECTION

Iterates over the elements of a collection in parallel.

SYNTAX

```
status parallel_FOREACH_IN_COLLECTION
[-checks]
itr_var
collections
body
```

Data Types

<i>itr_var</i>	string
<i>collections</i>	list
<i>body</i>	string

ARGUMENTS

-checks	Enables the variable checks.
itr_var	Specifies the iterator variable. During each iteration, <i>itr_var</i> is set to a collection of exactly one object.
collections	Specifies a list of collections over which to iterate.
body	Specifies a script to execute one time per iteration at a worker process.

DESCRIPTION

Similarly to the **foreach_in_collection** command, the **parallel_FOREACH_IN_COLLECTION** command iterates over elements of a collection, but with the added constraints and benefits of parallelism. It improves core utilization during the execution of custom Tcl scripts by executing multiple independent iterations simultaneously at worker processes. The required arguments for the **parallel_FOREACH_IN_COLLECTION** command are the same as the arguments for the **foreach_in_collection** command: an iterator variable, the collections over which to iterate, and a script to apply at each iteration.

The **parallel_FOREACH_IN_COLLECTION** command

- Automatically gathers any output produced in parallel at workers during the execution of the *body* script. It prints the output, sorted by iteration order at the master. For best performance, keep printing statements to a minimum as printing is done as a serial process.

- Requires a linked design; it attempts to perform an automatic link if a design is loaded but not linked.
- Supports the built-in **break**, **continue**, **return**, and **exit** commands, with the same syntax as other standard loop iteration constructs.
- Honors the resource constraints specified by using the **set_host_options** command with the **-local_process** and **-max_core** options.
- Falls back to running sequentially if used in the master in distributed multi-scenario analysis (DMSA) mode.
- Falls back to running sequentially if nested within another parallel command such as **redirect -bg**, **parallel_execute**, or **parallel_foreach_in_collection** itself.
- Falls back to running sequentially if used while parasitics are being read in the background.

Constraints

The *body* script argument is subject to the following constraints, some of which can be lifted by using the sibling command **post_eval**:

- Modifications cannot be performed on any collections specified by the *collections* argument (for example, modifications through the use of the **append_to_collection** command).
- Modifications cannot be performed on the iterator variable *itr_var* during the loop.
- The *body* script can read and modify variables and call procedures defined in the local or global scope. However, it is forbidden to do so in a way that would lead to dependencies between iterations. In particular, an iteration of *body* is not allowed to write to a variable from local or global scope if another iteration reads the same variable.

Note: The *body* script is run in parallel, and dependencies between iterations can potentially cause incorrect or unpredictable results due to race conditions.

- The *body* script is subject to the same restrictions imposed on the *script* argument of the **parallel_execute** command; that is, no change to the design state is allowed. This includes editing the design, changing constraints, performing updates, and setting user-defined attributes. If you attempt to execute such commands, the tool issues error messages.

Additionally, do not expect modifications to local and global variables from within the *body* script to persist after the loop is completed because the *body* script is executed at worker processes (likewise parallel_execute).

Sibling Command: post_eval

You can issue the **post_eval** command from within **parallel_foreach_in_collection** *body* to perform persistent changes to variables and the design state. The **post_eval** command has a single script argument that is submitted for execution at the master process. This script is executed sequentially, and in guaranteed iteration order.

For more information, see the man page of the **post_eval** command.

Recommended Usage

The **parallel_foreach_in_collection** command speeds up custom complex Tcl scripts by applying parallelism to collection iteration, both pre- and post-update.

Although most commands in PrimeTime are capable of utilizing multiple cores (such as **get_pins** and **get_timing_paths**), many commonly-used built-in Tcl commands in custom Tcl scripts are inherently sequential. Therefore, their execution utilizes only one CPU core even if there are multiple cores allocated to PrimeTime. The **parallel_foreach_in_collection** command improves performance and scalability by utilizing the available CPU cores.

To convert your scripts to use the **parallel_foreach_in_collection** command, identify existing **foreach_in_collection** loops for conversion with the following guidelines:

- The loop should have a large overall runtime and iterate over many objects. If the collection is too small, then an ideal work balance cannot be achieved, resulting in under-utilization of available CPU cores.
- The loop should not contain any heavyweight commands that completely change the design such as timing, noise, or power updates, or link_design. These commands are generally too compute-intensive to be in a loop, and they are forbidden in **parallel_foreach_in_collection**.
- Each iteration of the loop should have a relatively compute-intensive filtering, reporting, analysis, or decision-making component, followed by one or more simple components, such as setting an attribute or modifying a variable to aggregate the results. In general, these components map to the *body* and **post_eval** arguments, respectively.
- Lastly, each iteration of the loop should have reasonable memory requirements, because parallel execution of multiple iterations can increase total memory consumption proportionally to this increment.

After you identify a loop for conversion to **parallel_foreach_in_collection**, review and restructure the loop so that you can add the relevant **post_eval** statements at

the right places. Use the following guidelines for conversion:

- Global and local variables that are only read can be left unchanged, but variables that are modified need to be identified and classified according to their purpose. Temporary variables can be modified in the scope of the *body*, but any variable that is expected to persist after **parallel_foreach_in_collection** is completed needs to be moved into a **post_eval** statement.
- If there is any dependency between variables used in different iterations, it must be eliminated by rewriting the relevant part of the script. If a dependency cannot be removed, then the loop is no longer a candidate for conversion to **parallel_foreach_in_collection**.
- Any changes to the design state, setting of constraints or user-defined attributes must be moved into a **post_eval** statement. During this process, it is important to ensure that there are no dependencies on design changes across loop iterations.
- The resulting **post_eval** statements should be reviewed to make sure that they are kept as small and fast as possible. **post_eval** statements are executed serially and ultimately determine the scalability and performance of the whole loop.

Variable Checks

When variable checks are enabled with the **-checks** option, PrimeTime monitors access to all Tcl variables and detects a broad variety of problems related to the misuse of variables or violations of the recommended usage of **parallel_foreach_in_collection**.

The variable checks are designed to aid in migrating from **foreach_in_collection**; you still need to carefully review the Tcl script being converted.

There are four main categories of errors that can be issued by the variable checks. For more information about these errors, see the man pages for MC-201, MC-202, MC-203, and MC-204.

EXAMPLES

The following example computes the endpoint with most violators and the number of violators for this endpoint.

```
set endpoints [add_to_collection [all_registers -data_pins -async_pins]
                           [all_outputs] ]
set most_violators 0
set ep_with_most_violators ""
parallel_foreach_in_collection -checks endpoint $endpoints {
    set paths [get_timing_paths -to $endpoint -nw 1000
               -slack_lesser_than 0]
    set num_violators [size_of_collection $paths]
    post_eval {
```

```
if { $num_violators > $most_violators } {
    set most_violators $num_violators
    set ep_with_most_violators $endpoint
}
}
}
```

SEE ALSO

`foreach_in_collection(2)`
`parallel_execute(2)`
`post_eval(2)`
`redirect(2)`
`set_host_options(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)"
```

parse_proc_arguments

```

        }
    }
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).
```

post_eval

Submits a script for sequential execution in the context of a parallel iterator.

SYNTAX

```
status post_eval
script
```

Data Types

script string

ARGUMENTS

script
Specifies a script that to be executed sequentially at the master.

DESCRIPTION

The **post_eval** command can be used in the context of a parallel iterator such as the **parallel.foreach.in.collection** command to submit a script for execution at the master.

You can use multiple **post_eval** statements per loop iteration, and they can be put inside other constructs (such as **if** statements). **post_eval** statements are executed sequentially, and in guaranteed loop iteration order.

post_eval also supports built-in **break**, **continue**, **return**, and **exit** commands, with the same syntax as other standard loop iteration constructs.

post_eval lifts from some of the restrictions normally imposed on the *body* of parallel iterators, and therefore can be used to perform actions such as changing constraints, setting user-defined attributes, and modifying persistent variables to aggregate results.

However, **post_eval** is still forbidden from executing any heavyweight commands that completely change the design such as timing, noise, or power updates, and link_design. These commands are generally considered to be too compute-intensive to be in a loop in the first place. If you use these commands, the tool issues error messages.

It is an error to issue a **post_eval** command outside of the context of a parallel iterator.

Variable Transfer and Identification

The **post_eval** argument is parsed at the beginning of the loop to identify references to variables from the worker process. Only variable references beginning with a '\$' can be identified, and comments are ignored.

Because of this parsing requirement, **post_eval** statements can only be issued directly from within **parallel_foreach_in_collection** body (not from called Tcl procedures). Similarly, while the script argument to **post_eval** can call Tcl procedures, the parsing mechanism for variable identification does not follow called Tcl procedures.

The value of each of these variables at the worker at the point of the call to **post_eval** is sampled and transferred to the master so that, when the **post_eval** script is executed at the master, references to these variables can be successfully evaluated.

All basic Tcl types are supported for variable transfer, including string, boolean, integer, long, float, double, list, array, and dict. Collections of some types can also be transferred.

Supported collection object classes are pin, net, cell, port, lib, lib_cell, lib_pin, design, clock, and path_group.

For best performance, keep **post_eval** statements as small and fast as possible. It is important to avoid excessive variable transfer and only transfer small lists and arrays.

SEE ALSO

`collections(2)`
`parallel_foreach_in_collection(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 patterns 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.

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	0	7	2
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	0	12	0
PARA-004	Warning	100 *	1	0
APP-027	Information	100	150	50
APP-99	Information	0	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_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

Use this command to query 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]  
ref_name
```

Data Types

<i>domain_name</i>	string
<i>ref_name</i>	string

ARGUMENTS

```
-domain domain_name  
            Limits the search to the instances in the power domain specified.
```

```
ref_name  
            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_object** 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** command 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 port, return the name of the net connected to the specified port_name. If no net is connected, a null string is returned. This is UPF query command.

SYNTAX

list **query_port_net**

port_name
[-conn *connection_option_string*]

Data Types

port_name string
connection_option_string string

ARGUMENTS

port_name
Specifies the port name to query.

-conn [*low/high*]
Queries the LowConn or HighConn (the default) connection. This option can only be specified if *port_name* is not on a leaf cell.

DESCRIPTION

The **query_port_net** commands return the net logically connected to a port.

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_aocvm

Reads advanced on-chip variation (AOCV) derating factor tables.

SYNTAX

```
int read_aocvm
[-quiet]
aocvm_file
```

Data Types

aocvm_file string

ARGUMENTS

-quiet

Suppresses the SEL-004, SEL-005, and AOCVM-006 errors and warnings. If this option is not set, errors and warnings are generated when objects specified in the AOCV tables are not found in the current design or libraries.

aocvm_file

Specifies the name of the A OCV file.

DESCRIPTION

The **read_aocvm** command reads AOCV derate factor tables from a disk file. The tables are annotated onto one or more design objects. Allowed design object types are hierarchical cells, library cells and designs. The AOCV data is used to reduce pessimism and improve the accuracy of results.

When named AOCV table sets are specified on objects in the design using the **set_aocvm_table_group** command, a cell (or net) derives its AOCV deratings from the table set associated with the hierarchical cell (or design) that (a) fully contains the cell (or net), and (b) is the lowest level (closest ancestor) hierarchical cell (or design) with an associated AOCV table set. If no such hierarchical cell (or design) containing the cell (or net) is found, the unnamed AOCV table set is used. Note that a net inherits the AOCV derate set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating comes from an AOCV table set that contains the block and fully encloses the net.

When applying AOCV tables on multiple object types that apply to an arc, the following priority, in decreasing order of precedence, is used to determine the table that applies to the arc:

- 1) Library cell table
- 2) Hierarchical cell table
- 3) Design table

Note: Where a named AOCV table set applies to an arc, the precedence rules above apply to tables that belong to the specified set.

In PrimeTime version D-2010,06, the precedence rules above were different in that library cell tables had lower precedence than hierarchical cell tables. To use the previous behavior, set the **timing_derate_precedence_compatibility** variable to **true**.

Tables are processed in the order that they appear in the AOCV file. The last table of a specific type {object_type, rf_type, delay_type, derate_type} to be defined for an object in the file overwrites previous tables defined for the same object of the same type.

The **read_aocvm** command can read binary and compressed binary AOCV files created using the **write_binary_aocvm** command. No additional arguments are required to read binary or compressed binary AOCV files.

EXAMPLES

The following example shows an AOCV derate file, test.aocvm, with a single table annotated on all library cells.

```
pt_shell> sh cat test.aocvm

version:      1.0

object_type:  lib_cell
rf_type:      rise
delay_type:   cell
derate_type:  late
object_spec:  my_lib/*
voltage:     1.02
depth:        1 2 3
distance:    100 1000
table:        1.20 1.10 1.08
              1.22 1.15 1.11
```

```
pt_shell> read_aocvm test.aocvm
```

SEE ALSO

```
get_timing_paths(2)
remove_aocvm(2)
report_aocvm(2)
report_timing(2)
reset_aocvm_table_group(2)
set_aocvm_coefficient(2)
set_aocvm_table_group(2)
write_binary_aocvm(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_ddc

Reads in design files in the Synopsys DDC format.

SYNTAX

```
string read_ddc
[-netlist_only]
[-scenario scenario_name]
file_names
```

Data Types

<i>scenario_name</i>	string
<i>file_names</i>	list

ARGUMENTS

-netlist_only

Indicates that only the netlist is read; attributes are not read. This can greatly increase performance while reading designs.

-scenario scenario_name

Specifies the name of the **read_ddc** command scenario reads to get the attributes. The DDC file should be created using scenarios.

file_names

Specifies names of one or more files to be read. Typically, only one file is read.

DESCRIPTION

Reads design information from Synopsys DDC files into PrimeTime. DDC is a proprietary file format to store netlist information and constraint data. It is generated by Design Compiler, Physical Compiler XG mode, or IC Compiler.

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.

The DDC design files can contain many complex attributes. However, only certain attributes apply to PrimeTime (for example, clocks and other design constraints). By default, these attributes are loaded from the DDC file. However, all information can also be brought into PrimeTime through external scripts or other file formats (for example, SDC files). In these cases, use the **-netlist_only** option to read the DDC file.

EXAMPLES

In the following example, the file newcpu.ddc 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_ddc newcpu.ddc  
Loading ddc file '/designs/newcpu/v1.6/dbs/newcpu.ddc'  
Loaded 1 design.  
newcpu  
1
```

SEE ALSO

[read_db\(2\)](#)
[list_designs\(2\)](#)
[remove_design\(2\)](#)
[search_path\(3\)](#)

read_file

Reads a netlist or library file. This is a DC Emulation command provided for compatibility with Design Compiler.

SYNTAX

```
string read_file
[-format type]
[-single_file ns1]
[-names_file ns2]
[-define ns3]
file_list
```

Data Types

<i>type</i>	string
<i>ns1</i>	string
<i>ns2</i>	string
<i>ns3</i>	string
<i>file_list</i>	list

ARGUMENTS

```
-format type
        Specifies the file format. Allowed values are db, edif, vhdl, and verilog.
-single_file ns1
        Not supported by PrimeTime.
-names_file ns2
        Not supported by PrimeTime.
-define ns3
        Not supported by PrimeTime.
file_list
        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_edif(2)

```
read_verilog(2)
read_vhdl(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 reference manuals.

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_milkyway

Reads in one linked design from milkyway database.

SYNTAX

```
int read_milkyway
[-version version]
[-netlist_only]
[-library path]
[-scenario scenario_name]
CEL_name
```

Data Types

<i>version</i>	int
<i>path</i>	string
<i>scenario_name</i>	string
<i>CEL_name</i>	string

ARGUMENTS

-version *version*
Specifies the version of the design to be read. For example, there are design files under the CEL view in the Milkyway design library `design_lib`:
`\'design_lib/CEL/design1_pre_route1:1\'`
`\'design_lib/CEL/design1_post_route:2\'`
The 1 or 2 after the `:` is the version number of the design. The default is to read the most current version.

-netlist_only
Indicates that only the netlist is to be read; constraints are not read. The default is to read both netlist and constraints.

-library *path*
Specifies the absolute or relative path to the Milkyway design library. This option can be left out if the `mw_design_library` variable specifies the path to the Milkyway design library.

-scenario *scenario_name*
The Milkyway database is capable of storing multiple constraints that can correspond to various scenarios of running the design. This option specifies the name of the scenario for reading in constraints from Milkyway database. The default is to not use a scenario.

CEL_name
Specifies the design file name to be read. For example, there are design files under the CEL view in the Milkyway design library `design_lib`:
`\'design_lib/CEL/design1_pre_route1:1\'`
`\'design_lib/CEL/design1_post_route:2\'`
The `design1_pre_route` or `design1_post_route` are the *CEL_name* argument. Do not include version number in this argument.

DESCRIPTION

Reads a linked design from Milkyway database into PrimeTime. The command can also optionally load the timing constraints from the Milkyway database.

The **read_milkyway** command always reads in a linked design. Conceptually it takes the place of the usual read / link sequence in PrimeTime, and so the client must set the **link_path** and **search_path** variables before invoking the command. If there is already a linked design when the **read_milkyway** command is invoked, the existing design is unlinked before the Milkyway design is read in. This is consistent with the way the **link_design** command works.

A **link** command following a **read_milkyway** command is not necessary and should not be used because it could invalidate certain types of cell hierarchy. This method of reading Milkyway designs is consistent with other tools, but is different from reading Verilog or VHDL designs, which require a link.

The **read_milkyway** command can be affected by the values of the following variables:

- **link_path**: This is used to perform the implicit link during the **read_milkyway** command.
- **link_path_per_instance**: If this variable is set so that certain instances have special link_path, that variable setting is honored even in the **read_milkyway** flow.
- **search_path**: This is used for autoloading the technology libraries.
- **mw_design_library**: If the library option is missing from the command line then **read_milkyway** gets the library path from this variable. If the library option is present then the variable is set to the value of the library option.
- **mw_logic0_net**: Use this name for logic 0 constant nets. The default is "VSS".
- **mw_logic1_net**: Use this name for logic 1 constant nets. The default is "VDD".

The **link_create_black_boxes** variable does not affect the functional capability of the **read_milkyway** command. This is because the **read_milkyway** command reads a linked design. If an expected library cell is not present in the libraries specified by the **link_path** or **link_path_per_instance** variable, a black box is created irrespective of the value of the **link_create_black_boxes** variable.

EXAMPLES

In the following example, the latest version of the mw_des design is read into PrimeTime from the mw_db Milkyway database. Any nets set to a constant 0 are created as "VSS", and any set to a constant 1 are created as "VDD".

```
pt_shell> read_milkyway -netlist_only -library ./mw_db mw_des
1
```

SEE ALSO

`list_designs(2)`
`remove_design(2)`
`link_path(3)`
`link_path_per_instance(3)`
`mw_design_library(3)`
`mw_logic0_net(3)`
`mw_logic1_net(3)`
`search_path(3)`

read_parasitics

Reads net parasitics information from an SPEF, DSPF, RSPF, PARA, or binary parasitics file (SBPF) 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

<i>file_fmt</i>	string
<i>completion_type</i>	string
<i>prefix</i>	string
<i>factor</i>	float
<i>ttype</i>	string
<i>file_name</i>	string
<i>xlist</i>	string
<i>ylist</i>	string
<i>flip_list</i>	string
<i>rot_list</i>	string
<i>file_names</i>	string

ARGUMENTS

-format *file_fmt*

Specifies the format of the parasitics file. The allowed values are SPEF, DSPF, RSPF, PARA (Milkyway), and Synopsys Binary Parasitics Format (SBPF). If the **-format** option is not specified, the application can determine whether the file is SBPF, 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 SBPF 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 and the SBPF 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. By 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 SBPF 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 SBPF 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-offset 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 SBPF, 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 Synopsys Binary Parasitics (SBPF) 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.

SBPF is an efficient format for sharing parasitics among Synopsys applications. You can use the PrimeTime **write_parasitics** command to write a parasitics file in SBPF.

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 SBPF format. Resistances connected to the ground are not connected. The detailed RC network must be complete; incomplete RC networks are discarded and the delays are computed using wire-load

models. You can specify detailed parasitics with the SPEF, SBPF, 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.

Annotated parasitics are used to compute cell delays and net delays with more accuracy. Because the delay calculation involves the computation of the actual waveforms of the transitions, you must provide PrimeTime with a way to interpret the delays from the Synopsys library. The definitions of the delays and transition times is communicated to PrimeTime with environment variables, which specify the different characterization trip-points used when generating the Synopsys library. For example, these trip-points can specify that for a rising cell delay, the delay is defined as the delay between the time that the input transition reaches 50 percent of the voltage source, and the time that the output transition reaches 50 percent of the voltage source. Similarly, the trip-points can specify that a rising transition time is defined as the delay between the time that the transition reaches 20 percent of the voltage source and the time that the transition reaches 80 percent of the voltage source. You specify these trip-points in libraries.

You can turn on the capacitive crosstalk effect on the delay by setting the **si_enable_analysis** variable to **true**. You can filter some of the coupling capacitances by specifying the following variable:

Variable Name	Default
si_filter_per_aggr_noise_peak_ratio	0.0

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.

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 SBPF.

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.spf
pt_shell> read_parasitics -pin_cap_included -verbose mydesign.spf
```

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.spf
```

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.spf
```

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]
[-path prefix]
[-ignore ignore_name]
[-exclude exclude_file_name]
[-derate_glitch value]
[-quiet]
```

Data Types

file_name	string
prefix	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.

-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

<i>sdc_version</i>	string
<i>file_name</i>	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)
```

read_sdc

524

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 load_delay_type]
[-analysis_type type]
[-min_file min_fname]
[-max_file max_fname]
[-type sdf_delay_type]
[-min_type min_delay_type]
[-max_type max_delay_type]
[-cond_use use_type]
[-syntax_only]
[-strip_path path_name]
[-verbose]
[-worst]
file_name
```

Data Types

<i>path_name</i>	string
<i>load_delay_type</i>	string
<i>type</i>	string
<i>min_fname</i>	string
<i>max_fname</i>	string
<i>sdf_delay_type</i>	string
<i>min_delay_type</i>	string
<i>max_delay_type</i>	string
<i>use_type</i>	string
<i>path_name</i>	string
<i>file_name</i>	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 *load_delay_type*
Indicates whether load delays are included in net or cell delays in the timing file being read. The available values are the default of *cell* and also *net*. The load delay is the portion of cell delay arising from the capacitive load of the net driven by the cell.

-analysis_type *type*
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_delay_type

Indicates which of the SDF triplet delay values, *sdf_min*, *sdf_typ*, or *sdf_max*, are read from the SDF file. Delays in SDF are represented in the form of triplets (*sdf_min:sdf_typ:sdf_max*). By default, The **read_sdf** command reads the maximum delays *sdf_max* value.

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 min_delay_type

Specifies which of the SDF triplet delay values, *sdf_min*, *sdf_typ*, or *sdf_max*, are read from the SDF file for minimum delay. Delays in SDF are represented in the form of triplets (*sdf_min:sdf_typ:sdf_max*). By default, the **read_sdf** command reads the minimum delays *sdf_min* value. Use this option only with the *-analysis_type bc_wc / on_chip_variation* option.

-max_type max_delay_type

Specifies which of the SDF triplet delay values, *sdf_min*, *sdf_typ*, or *sdf_max*, are read from the SDF file for maximum delay. Delays in SDF are represented in the form of triplets (*sdf_min:sdf_typ:sdf_max*). By default, the **read_sdf** command reads the maximum delays *sdf_max* value. Use this option only with the *-analysis_type bc_wc / on_chip_variation* option.

-cond_use use_type

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
 Indicates that no timing annotation performed; only syntax is processed. Use this option to verify that your SDF syntax is correct and will not issue any error messages.

-strip_path path_name
 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** command 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_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

<i>pipe</i>	string
<i>path</i>	string
<i>strip_path</i>	string
<i>window_list</i>	list
<i>condition</i>	string
<i>file_name</i>	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 the 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 **-interval** value in the **create_power_waveforms** command. For example, if the **-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 are 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 Verilog files.

SYNTAX

```
string read_verilog
[-hdl_compiler]
file_names
```

Data Types

file_names list

ARGUMENTS

-hdl_compiler

Indicates that the Verilog files are to be read using the PrimeTime external reader (ptxr) that uses HDL Compiler. Reading files in this way requires an HDL Compiler license while the read is in progress. HDL Compiler supports the complete Verilog language, but uses more CPU and memory than does the native PrimeTime Verilog reader.

file_names

Specifies the names of one or more files to be read.

DESCRIPTION

Reads one or more structural, gate-level Verilog netlists into PrimeTime. 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 were no **search_path**. To determine the file that the **read_verilog** command loads, use the **which** command. After the Verilog files are loaded, to view the design objects, use the **list_designs** command. To remove designs, use the **remove_design** command.

The Verilog netlist must contain fully-mapped, structural designs. PrimeTime cannot link or perform timing analysis with netlists that are not fully mapped at the gate level. There must be no Verilog high-level constructs in the netlist.

By default, the **read_verilog** command invokes a native PrimeTime Verilog reader. For large netlists, this reader is much faster and more memory-efficient than HDL Compiler, but is limited to structural Verilog. The file can be a simple ASCII file, or it can be compressed with gzip. The native Verilog reader automatically detects all of this format information from the file. Files that are compressed with gzip are first uncompressed to a temporary file in the directory stored in the **pt_tmp_dir** variable.

Generally, the native Verilog reader does not create unconnected nets. If you need to have these nets in your design, set the **svr_keep_unconnected_nets** variable to **true**.

If you have an HDL Compiler license and prefer to use HDL Compiler, you can invoke it from PrimeTime using the **read_verilog -hdl_compiler** command. When reading Verilog with a PrimeTime external reader (in this case, HDL Compiler), the procedure for interrupting the **read_verilog** command is slightly different than for most PrimeTime commands. Normally, one Control-c terminates a command, and three Ctrl+C entries in sequence terminates PrimeTime. However, to terminate the **read_verilog** command, enter Control-c three times. This terminates the read process but it does not terminate PrimeTime. This is especially useful if you want to terminate the reading of a very long netlist file.

You can use the **bus_naming_style** variable to control bus naming, but setting this variable in PrimeTime does not affect the reading of Verilog files with HDL Compiler.

Limitations of the Native Verilog Reader in PrimeTime

The native Verilog reader in PrimeTime has the following limitations:

- No non-structural constructs are supported. For more information, see the "Supported Language Subset for Native Verilog Reader in PrimeTime" section.
- Parameters are not supported. The parameter and defparam statements are read and ignored.
- Gate instantiations are not supported.

Supported Language Subset for the Native Verilog Reader in PrimeTime

The native Verilog reader supports a small structural subset of the Verilog language. Additional constructs are recognized and ignored. Supported constructs are as follows:

- module/endmodule
- input, inout, and output
- wire
- tri, wor, and wand: These constructs are supported, but are considered to be the same as the wire construct. That is, there is no special significance to these constructs.
- supply0 and supply1: These create wires that are logic 0 and 1, respectively.

- assign
- tran: An exception from the gate instantiation subset, tran is supported to the extent that it relates one wire to another, as with assign. Beyond that, tran has no special significance.
- The preprocessor directives 'define, 'undef, 'include, 'ifdef, 'else, and 'endif are not supported by default, as the target for this reader is structural, machine generated Verilog, which rarely contains these constructs. You can enable a preprocessor phase by setting the **svr_enable_vpp** variable to **true**. This makes a pre-pass over the file looking for and expanding these constructs, outputting a temporary file in the directory stored in the **pt_tmp_dir** variable.
- All simulation directives (for example, 'timescale) are recognized and ignored.
- The specify/endspecify construct and its contents are recognized and ignored.
- Like HDL Compiler, the comment directives **translate_off** and **translate_on** are used to bypass Verilog features not supported by the reader. The following example shows how to bypass an unsupported feature:

```
// Synopsys translate_off
nand (n2, a1, s2);
/* Synopsys translate_on */
```

Limitations of the HDL Compiler

Reading with the **-hdl_compiler** option has these limitations:

- Any variables that affect Verilog reading are read from .synopsys_dc.setup files, not from .synopsys_pt.setup. These variables cannot be set from within PrimeTime. You can set the **ptxr_setup_file** variable in PrimeTime to restrict ptxr so that it reads only the system setup file and ignores your home and local setup files.
- Some of the error messages that appear from the **read_verilog** process cannot be found using the **man** command from PrimeTime. You can access these message man pages from other Synopsys shells or from the UNIX man command.
- Pragmas are ignored.
- No warnings are issued if unmapped logic or HDL constructs exist in the Verilog netlist.

EXAMPLES

In the following example, the file newcpu.v 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_verilog newcpu.v
Loading verilog file '/designs/newcpu/v1.6/newcpu.v'
1
pt_shell> remove_design -all
Removing design newcpu...

pt_shell> read_verilog newcpu.v -hdl_compiler
Beginning read_verilog...
Loading db file '/release/libraries/syn/standard.sldb'
Loading db file '/release/libraries/syn/gtech.db'
Loading verilog file '/designs/newcpu/v1.6/newcpu.v'
Reading in the Synopsys verilog primitives.
/designs/newcpu/v1.6/newcpu.v:
1
```

SEE ALSO

```
list_designs(2)
remove_design(2)
which(2)
bus_naming_style(3)
pt_tmp_dir(3)
ptxr_setup_file(3)
search_path(3)
svr_enable_vpp(3)
svr_keep_unconnected_nets(3)
```

read_vhdl

Reads in one or more VHDL files.

SYNTAX

```
status read_vhdl
[-vhdl_compiler]
file_names
```

Data Types

file_names list

ARGUMENTS

-vhdl_compiler

Indicates that the VHDL files are to be read using VHDL Compiler. Reading files in this way requires a VHDL Compiler license while the read is in progress.

file_names

Specifies names of one or more files to be read.

DESCRIPTION

Reads one or more structural, gate-level VHDL netlists 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**. Files that have absolute path names are loaded as though there is no **search_path**. To determine the file that the **read_vhdl** command loads, use the **which** command. After the VHDL files are loaded, to view the design objects, use the **list_designs** command. To remove designs, use the **remove_design** command.

The VHDL netlist must contain fully-mapped, structural designs. PrimeTime cannot link or perform timing analysis with netlists that are not fully mapped at the gate level. There must be no VHDL high-level constructs in the netlist.

By default, the **read_vhdl** command uses the same VHDL netlist reader as Design Compiler. This reader does not require any licenses. If you have a VHDL Compiler license and prefer to use VHDL Compiler, you can invoke it from PrimeTime using the **read_vhdl -vhdl_compiler** command.

The procedure for interrupting the **read_vhdl** command is slightly different than most PrimeTime commands. Normally, one Control-c terminates a command, and three Control-c entries in sequence terminates PrimeTime. However, to terminate the **read_vhdl** command, enter Control-c three times. This terminates the read process but does not terminate PrimeTime. This is especially useful if you want to terminate the reading of a very long netlist file.

Reading with the **read_vhdl** command has these limitations:

- Any variables that affect VHDL reading are read from .synopsys_dc.setup files, not from .synopsys_pt.setup. These variables cannot be set from within PrimeTime. You can set the **ptxr_setup_file** variable in PrimeTime to restrict ptxr so that it reads only the system setup file and ignores your home and local setup files.
- To support reading of entity/architecture pairs from separate files, all files in the *file_names* list are read at once. If the VHDL files you are reading are unrelated, you should read them in separate **read_vhdl** commands. This avoids memory peaks in either ptxr or PrimeTime.
- Some of the error messages that appear from the **read_vhdl** process cannot be found using the **man** command from PrimeTime. You can access these message man pages from other Synopsys shells or from the UNIX man command.
- Pragmas are ignored.
- No warnings are issued if unmapped logic or HDL constructs exist in the VHDL netlist.

EXAMPLES

In the following example, the file newcpu.vhd 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_vhdl newcpu.vhd
Beginning read_vhdl...
Loading vhdl file '/designs/newcpu/v1.6/newcpu.vhd'
Performing 'read' command.
New VHDL reader completed successfully.
end
1
```

SEE ALSO

```
list_designs(2)
remove_design(2)
which(2)
ptxr_setup_file(3)
search_path(3)
```

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

<i>target</i>	string
<i>command_string</i>	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 } {

redirect
```

```
    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)
```

remote_execute

Builds a buffer of commands and triggers execution of these commands on remote slaves.

SYNTAX

```
status remote_execute
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-verbose]
command_string
```

Data Types

<i>pre_command_string</i>	string
<i>post_command_string</i>	string
<i>command_string</i>	string

ARGUMENTS

-pre_commands *pre_command_string*
Specifies a string of commands to be executed in the slave context before the execution of the default command string. Commands are grouped using the semicolon ";" character.

-post_commands *post_command_string*
Specifies a string of commands to be executed in the slave context after the execution of the default command string. Commands are grouped using the ";" character.

-verbose
Specifies that the slave output should be piped back to the master console.

command_string
Specifies a string of commands to be executed in the slave context. To group commands, use the semicolon ";" character.

DESCRIPTION

The **remote_execute** command builds up a buffer of commands and triggers execution of these commands on remote slaves. All commands in the buffer are executed in the order in which they enter the buffer. This command can only be executed when you have set the current session. Use curly braces {} to force variables and expressions to be evaluated in a slave context. Use quotation marks (" ") to force variables and expressions to be evaluated in a master context.

Multi-Scenario Analysis Mode

In the multi-scenario analysis mode, the very first invocation of the **remote_execute** command for a particular session triggers baseline image generation.

Use the **-pre_commands** and **-post_commands** options to specify commands to be executed before and after the commands being remotely executed.

EXAMPLES

In the following examples, the s1 scenario is in focus. In the following example, the **remote_execute** command is used to execute a nonmerged report_timing. The output of the **report_timing** command is found in the \$multi_scenario_working_directory/s1/out.log file.

```
pt_shell> remote_execute {report_timing}

Start of Master/Slave Task Processing
-----
Started : Command execution in scenario 's1'
Command execution in scenario 's1' : Succeeded
Finished : Command execution in scenario 's1'
-----
End of Master/Slave Task Processing
1
```

In the next example, the situation is as in the first example except that the **-pre_commands** option is used to set a variable before the report.

```
pt_shell> remote_execute -pre_commands {set
rc_slew_lower_threshold_pct_fall 30.0} {report_timing}

Start of Master/Slave Task Processing
-----
Started : Command execution in scenario 's1'
Command execution in scenario 's1' : Succeeded
Finished : Command execution in scenario 's1'
-----
End of Master/Slave Task Processing
1
```

SEE ALSO

[report_multi_scenario_design\(2\)](#)

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

<i>from_pins</i>	list
<i>to_pins</i>	list
<i>clock_check</i>	string
<i>object_spec</i>	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

```
string remove_annotated_clock_network_power
[-clock domain_clock]
```

Data Types

clock_object string

ARGUMENTS

-clock *domian_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.

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

set_annotated_clock_network_power(2)
report_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

<i>from_list</i>	list
<i>to_list</i>	list
<i>object_spec</i>	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_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

Remove previously annotated power from unresolved blackbox cells or leaf cells.

SYNTAX

```
int remove_annotated_power
-all
cell_list
```

Data Types

cell_list list

ARGUMENTS

-all

Indicates 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 powers previously annotated on cells using the **set_annotated_power** command. Both the internal power and the leakage power annotated by the **set_annotated_power** command are to be removed. There is no way to remove only internal power or leakage power. You can remove all annotated power by using the *-all* option or remove only the power annotated on the specified cell list.

EXAMPLES

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -int 1 -leak 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)
```

Cell type	Total	Annotated	NOT 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
*****
```

Cell type	Total	Annotated	NOT Annotated
unresolved black-box cell	2	1	1
leaf cell	3	0	3
	5	1	4

1

SEE ALSO

`set_annotated_power(2)`
`report_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_annotated_transition
562
```

remove_aocvm

Removes advanced on-chip variation (AOCV) information.

SYNTAX

```
status remove_aocvm
[-coefficient]
[-derate]
[object_list]
```

Data Types

object_list list

ARGUMENTS

-coefficient

Removes only the AOCV coefficients. The **-coefficient** and **-derate** options are mutually exclusive.

-derate

Removes only the AOCV derating factors. The **-coefficient** and **-derate** options are mutually exclusive.

object_list

Specifies the objects from which AOCV information is removed.

DESCRIPTION

Removes user-specified AOCV coefficients and derating factors that were previously set with the **set_aocvm_coefficient** and **read_aocvm** commands, respectively. If the command is specified with no options, all AOCV data is removed.

To display AOCV deratings and coefficients, use the **report_aocvm** command.

EXAMPLES

The following example removes the AOCV coefficients from the library cell named lib/bufA:

```
pt_shell> remove_aocvm -coefficient lib/bufA
```

The following example removes all AOCV derating factors that were set using the **read_aocvm** command:

```
pt_shell> remove_aocvm -derate
```

The following example removes all AOCV derating factors from the current design:

```
pt_shell> remove_aocvm -derate [current_design]
```

SEE ALSO

`read_aocvm(2)`
`report_aocvm(2)`
`set_aocvm_coefficient(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 multi-output 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

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)
PARA-061
```

remove_capacitance

Removes capacitance on nets or ports.

SYNTAX

```
string 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_internal_loads(2)
report_port(2)
reset_design(2)
set_drive(2)
```

```
set_wire_load(2)
```

remove_capacitance
568

remove_case_analysis

Removes the case analysis value on input.

SYNTAX

```
string remove_case_analysis
port_or_pin_list
```

Data Types

```
port_or_pin_list      list
```

ARGUMENTS

```
port_or_pin_list
Lists ports or pins for which the case analysis entry is to be removed.
```

DESCRIPTION

This command removes previously specified entries of case analysis on ports or pins. Case analysis is specified using the **set_case_analysis** command. The command specifies that a pin or port is at a constant logic value (0 or 1), or that only one of the rising or falling transition is valid.

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  
cell_list | -all
```

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.

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
574

remove_clock_gating_check

Captures clock-gating checks.

SYNTAX

```
string remove_clock_gating_check
[-setup]
[-hold]
[-rise]
[-fall]
[-high]
[-low]
[object_list]
```

Data Types

object_list list

ARGUMENTS

-setup
Indicates the removal of 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
Indicates the removal of 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
Indicates the removal of 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
Indicates the removal of 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
Remove 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 for 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.

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]
```

An alternative way to remove information set by the **set_clock_gating_check** command is to use the **reset_design** command.

SEE ALSO

```
set_clock_gating_check(2)
set_disable_clock_gating_check(2)
remove_disable_clock_gating_check(2)
current_design(2)
report_constraint(2)
reset_design(2)
```

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

SYNTAX

```
Boolean 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. You should use the **set_clock_groups** command to predefine these names. Substitute the list you want for the *name_list* variable. 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

Removes specific exclusive or asynchronous clock groups from the current design.

These clock groups are specified by the *name_list* variable from the current design. You must specify either the *-exclusive* or *-asynchronous* option. You must specify either the *name_list* or *-all* option.

To find the names for existing groups, use 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

<i>clock_list</i>	list
<i>object_list</i>	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]
[object_list]
```

Data Types

<i>from_clock</i>	list
<i>rise_from_clock</i>	list
<i>fall_from_clock</i>	list
<i>to_clock</i>	list
<i>rise_to_clock</i>	list
<i>fall_to_clock</i>	list
<i>object_list</i>	list

ARGUMENTS

-from *from_clock* -to *to_clock*

These two options specify the source and 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_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.

-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.

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.

-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

```
int 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 the timing context information.

SYNTAX

```
string remove_context
[-timing]
[-environment]
[-design_rules]
[-constant_inputs]
cell_list
```

Data Types

cell_list list

ARGUMENTS

-timing
Deletes timing-related context information such as clocks, input and output delays, and timing exceptions.

-environment
Deletes 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
Deletes characterized design rule checks such as max_capacitance, max_transition, and max_fanout.

-constant_inputs
Deletes characterized logic constants propagated to input pins.

cell_list
Provides a list of instances for which the context is deleted.

DESCRIPTION

Deletes the timing context captured using the **characterize_context** command for the specified list of instances. Options specify the categories of context information to delete. All context information is deleted if no option is specified.

EXAMPLES

The following command deletes the timing-related context information for instances I1 and I2.

```
pt_shell> remove_context -timing {I1 I2}
```

The following command deletes the environment and design rule context information for instance I2.

```
pt_shell> remove_context -env -design_rules I2
```

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

```
int remove_coupling_separation  
[-pairwise pnets]  
-all  
nets
```

Data Types

<i>pnets</i>	list
<i>nets</i>	list

ARGUMENTS

-pairwise *pnets*

When **-pairwise *pnets*** is applied, 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

When **-all** option is used, coupling separation constraints are reset on all the nets.

nets

Specifies a list of nets from which separation constraints are to be removed.

DESCRIPTION

This command removes the constraints set by **set_coupling_separation** command.

When **-pairwise *pnets*** is applied, only separation constraints between *pnets* and *nets* are removed.

Any prior settings on *pnets* or *nets* from the **set_si_delay_analysis** or **set_si_noise_analysis** command are restored.

Instances of this command are recorded for output by the **write_changes** command. This feature allows the user to communicate the separation constraint to other implementation tools along with netlist changes.

The **remove_coupling_separation** command returns a **1** if successful and a **0** if unsuccessful. If the remove command is used to remove a constraint that has not been set, the XTALK-107 warning message is issued.

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

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
```

To verify if the separation constraint was removed on the nets, use the **report_si_delay_analysis** or **report_si_noise_analysis** command with the *-coupling_separated* option.

SEE ALSO

```
set_coupling_separation(2)
report_si_delay_analysis(2)
report_si_noise_analysis(2)
write_changes(2)
```

remove_current_session

Removes the previously set session.

SYNTAX

Boolean **remove_current_session**

DESCRIPTION

In the multi-scenario analysis, a list of scenarios is brought into session and command focus (selected for analysis) by using the **current_session** command. These scenarios are then defocused using the **remove_current_session** command. However, the scenarios are not removed from memory and could be set in focus for another subsequent session.

EXAMPLES

In the following example, two scenarios are in the current session. These scenarios are then removed from focus.

```
pt_shell> current_session {scen2 scen3}
1
pt_shell> remove_current_session
1
```

SEE ALSO

`report_multi_scenario_design(2)`
`remove_scenario(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]
```

Data Types

<i>from_object</i>	object
<i>to_object</i>	object
<i>clock_object</i>	object

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.

-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_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.

-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

Removes one or more designs from memory.

SYNTAX

```
string remove_design
-all
-hierarchy
designs
```

Data Types

designs list

ARGUMENTS

-all

Indicates that all designs are to be removed. The **-all**, **-hierarchy**, and *designs* options are mutually exclusive; you can specify only one.

-hierarchy

Indicates that all designs in the hierarchy of the current design are to be removed, including the current design. The **-all**, **-hierarchy**, and *designs* options are mutually exclusive; you can specify only one.

designs

Specifies a list of designs to remove. The **-all**, **-hierarchy**, and *designs* options are mutually exclusive; you can specify only one.

DESCRIPTION

The **remove_design** command removes designs from pt_shell. You must specify either the **-all**, **-hierarchy** or *designs* option, but not more than one of them. The **remove_design** command does not remove libraries; use the **remove_lib** command for that purpose.

Note that you cannot combine the **-hierarchy** option with a list of designs. The **-hierarchy** option is restricted to the current design. If the current design is not linked, it is automatically linked. Then, all designs in the hierarchy of the current design are removed, including the current design itself.

EXAMPLES

The following example removes the 'd1' and 'd2' designs:

```
pt_shell> remove_design {d1 d2}
Removing design 'd1'...
Removing design 'd2'...
1
```

The following example removes all designs in memory:

```
pt_shell> remove_design -all
Removing design 'd3'...
1
```

The following example removes all designs in the hierarchy of the current design. In this example, the design was already linked.

```
pt_shell> remove_design -hierarchy
Removing design 'TOP'...
Removing design 'eBlock'...
Removing design 'iBlock'...
Removing design 'mBlock'...
1
```

SEE ALSO

`current_design(2)`
`list_designs(2)`
`remove_lib(2)`

remove_design_mode

Removes specified design modes and/or cell mode and path mappings to design modes.

SYNTAX

```
string remove_design_mode
mode_list
[-from from_pin_list]
[-to   to_pin_list]
[-through through_pin_list]
[cell_mode_list]
[instance_list]
```

Data Types

<i>mode_list</i>	list
<i>from_pin_list</i>	list
<i>to_pin_list</i>	list
<i>through_pin_list</i>	list
<i>cell_mode_list</i>	list
<i>instance_list</i>	list

ARGUMENTS

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.

-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 **design_mode** command 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 **design_mode** command 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 once 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.

cell_mode_list

Specifies a list of cell modes which are unmapped from the **design_mode command**. 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 command.
```

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 mode_list** 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 -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

```
map_design_mode(2)
define_design_mode_group(2)
report_mode(2)
set_mode(2)
reset_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 and pins 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
```

SEE ALSO

set_disable_clock_gating_check(2)

remove_disable_timing

Enables the previously disabled timing arcs.

SYNTAX

```
string remove_disable_timing
[-from from_pin_name]
[-to to_pin_name]
object_list
```

Data Types

<i>from_pin_name</i>	string
<i>to_pin_name</i>	string
<i>object_list</i>	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_lib(2)
set_disable_timing(2)
report_lib(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
*****
```

Input Port	Resistance		Transition	
	Rise	Fall	Rise	Fall
<hr/>				
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  
*****
```

Input Port	Resistance		Transition	
	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

<i>clock_name</i>	string
<i>port_list</i>	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

```
string 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)
set_min_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

<i>collection1</i>	collection
<i>object_spec</i>	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 *multiplex_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 specified 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.

EXAMPLES

In the following example, the master process is running on the host named ptopt021 using a 64-bit binary and specifies five different sets of host options.

The first host options, named "my_opts1", specifies 1 process with the same architecture as the master on the same host as the master and does not specify an upper limit on the number of cores to use; therefore, the default applies.

The second host options, named "my_opts2", specifies 2 processes on the same host as the master but explicitly mentions the name of the master's host and specifies to use a maximum of 2 cores per process.

The third host options, named "my_opts3", specifies 4 processes (with the same architecture as the master) on the host named ptopt010 using "rsh" to connect to ptopt010 and specifying to use a maximum of 3 cores per process.

The fourth host options named "my_opts4", specifies 5 processes (with the same architecture as the master) on an LSF farm, requesting 500MB of memory and 2 slots per compute server and specifying to use a maximum of 2 cores per process. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use hence the "-n 2 -R span\[ptile=2\]". A command is also provided for the termination of jobs that do not come online.

The fifth host options, named "my_opts5", specifies 2 processes (with the same architecture as the master) on a Grid farm requiring the jobs to be launched using the project named "bnormal" and does not specify an upper limit on the number of cores to use, hence the default will apply. A command is also provided for the termination of jobs that do not come online. Note: The "-b y" option is required when accessing grid.

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
1
pt_shell> set_host_options -name my_opts2 -num_processes 2 \
ptopto18
1
pt_shell> set_host_options -name my_opts3 -num_processes 4 \
-submit_command "/usr/bin/rsh -n" ptopto10
1
pt_shell> set_host_options -name my_opts4 -num_processes 5 \
-submit_command "/lsf/bin/bsub -n 2 -R rusage\[mem=500\] -R span\[ptile=2\]" \
-terminate_command "/lsf/bin/bkill"
1
pt_shell> set_host_options -name my_opts5 -num_processes 2 \
-submit_command "/grid/bin/qsub -b y -P bnormal" \
-terminate_command "/grid/bin/qdel"
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
1] Launched : ...
...
      Status : Forking successful
      Stdout : Process group is (31811)
      Stderr : **<<EMPTY>>**

2] Launched : ...
...
      Status : Forking successful
      Stdout : Process group is (31846)
      Stderr : **<<EMPTY>>**

3] Launched : ...
...
      Status : Forking successful
      Stdout : Process group is (31891)
```

```
Stderr  : **<<EMPTY>>**  
  
4] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Process group is (17082)  
    Stderr  : **<<EMPTY>>**  
  
5] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Process group is (17153)  
    Stderr  : **<<EMPTY>>**  
  
6] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Process group is (17231)  
    Stderr  : **<<EMPTY>>**  
  
7] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Process group is (17309)  
    Stderr  : **<<EMPTY>>**  
  
8] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Job <321813> is submitted to default queue <normal>.  
    Stderr  : **<<EMPTY>>**  
  
9] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Job <321814> is submitted to default queue <normal>.  
    Stderr  : **<<EMPTY>>**  
  
10] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Job <321815> is submitted to default queue <normal>.  
    Stderr  : **<<EMPTY>>**  
  
11] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Job <321816> is submitted to default queue <normal>.  
    Stderr  : **<<EMPTY>>**  
  
12] Launched : ...  
    ...  
    Status  : Forking successful  
    Stdout  : Job <321817> is submitted to default queue <normal>.  
    Stderr  : **<<EMPTY>>**
```

```

13] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188655 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**

14] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188668 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**


-----
-----  

Distributed farm creation timeout : 21600 seconds
Waiting for 14 (of 14) distributed hosts (Tue Nov 24 11:09:49 2009)
Waiting for 9 (of 14) distributed hosts (Tue Nov 24 11:09:59 2009)
Waiting for 7 (of 14) distributed hosts (Tue Nov 24 11:10:09 2009)
Waiting for 6 (of 14) distributed hosts (Tue Nov 24 11:10:19 2009)
Waiting for 4 (of 14) distributed hosts (Tue Nov 24 11:10:29 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:39 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:49 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:59 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:09 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:19 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:29 2009)
Waiting for 0 (of 14) distributed hosts (Tue Nov 24 11:11:39 2009)
-----  

-----
```

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: D-2010.06
Date   : Tue Jun 8 11:11:39 2009
*****
```

Options Name	Host	32Bit	Num Processes
my_opts1	**localhost**	N	1
my_opts2	**ptopt018**	N	2
my_opts3	ptopt010	N	4
my_opts4	>>farm<<	N	5
my_opts5	>>farm<<	N	2

remove_host_options

612

Options Name	#	Host Name	Job ID	Process ID	Status
<hr/>					
my_opts1	1	ptopt018	31811	31811	ONLINE
my_opts2	2	ptopt018	31846	31846	ONLINE
	3	ptopt018	31891	31891	ONLINE
my_opts3	4	ptopt010	17082	17082	ONLINE
	5	ptopt010	17153	17153	ONLINE
	6	ptopt010	17231	17231	ONLINE
	7	ptopt010	17309	17309	ONLINE
my_opts4	8	maddog131	321813	8373	ONLINE
	9	madyak036	321814	31490	ONLINE
	10	madyak726	321815	21995	ONLINE
	11	madcow037	321816	16787	ONLINE
	12	madyak040	321817	18278	ONLINE
my_opts5	13	peopf105	1188655	18524	ONLINE
	14	peopf33	1188668	8798	ONLINE

Hosts limits:

Name	Cores limits
	User
<hr/>	
** local process **	none
my_opts1	1
my_opts2	2
my_opts3	3
my_opts4	2
my_opts5	1

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 3
Shutdown Process 1
Shutdown Process 2
Shutdown Process 9
Shutdown Process 5
Shutdown Process 4
Shutdown Process 6
Shutdown Process 8
Shutdown Process 7
Shutdown Process 10
Shutdown Process 12
Shutdown Process 11
Shutdown Process 14
```

```
Shutdown Process 13
1
pt_shell> report_host_usage
*****
Report : host_usage
Version: D-2010.06
Date   : Tue Jun 1 11:11:40 2009
*****
```

Hosts limits:

Name	Cores limits	User
-----	-----	-----
** local process **		none

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

<i>object_list</i>	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

<i>clock_name</i>	list
<i>port_pin_list</i>	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_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

Removes a licensed feature.

SYNTAX

```
status remove_license
feature_list
```

Data Types

```
feature_list      list
```

ARGUMENTS

```
feature_list
```

Removes the licenses of the specified list of features. For a list of all features that are licensed at your site, see the key file.

DESCRIPTION

This command removes the specified licensed features from the features you currently obtain. The **remove_license** command is valid only when Network Licensing is enabled.

The **list_licenses** command provides a list of the features that you currently have checked out.

EXAMPLES

This example removes the STAMP Compiler and VHDL Compiler licenses.

```
pt_shell> remove_license {STAMP-Compiler VHDL-Compiler}
Checked in license 'Stamp-Compiler'
Checked in license 'VHDL-Compiler'
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_capacitance

remove_max_fanout

Removes maximum fanout limits from ports or designs.

SYNTAX

```
string 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)  
remove_min_fanout(2)  
report_constraint(2)  
report_design(2)  
report_port(2)  
get_ports(2)  
set_max_fanout(2)  
set_fanout_load(2)  
set_max_capacitance(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

```
string 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

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)  
remove_min_transition(2)  
report_constraint(2)  
report_design(2)  
report_port(2)  
get_ports(2)  
set_max_capacitance(2)  
set_max_fanout(2)
```

```
set_max_transition(2)
```

```
remove_max_transition  
632
```

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_capacitance  
634
```

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_multi_scenario_design

Removes all multi-scenario objects from memory and removes from disk all images generated by multi-scenario analysis.

SYNTAX

Boolean **remove_multi_scenario_design**

DESCRIPTION

The **remove_multi_scenario_design** command removes all multi-scenario objects from pt_shell. The current session and all scenarios are removed from memory. All netlist, baseline and current images generated during the multi-scenario analysis are removed from disk.

EXAMPLES

The following example removes all multi-scenario objects and removes all images from the disk.

```
pt_shell> remove_multi_scenario_design  
1
```

SEE ALSO

[report_multi_scenario_design\(2\)](#)

remove_net

Removes nets from the current design.

SYNTAX

```
int remove_net  
net_list | -all
```

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.

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

<i>clock_name</i>	string
<i>port_pin_list</i>	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

Boolean **remove_parasitic_corner**

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 not 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_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_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_transition

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

<i>attribute_name</i>	string
<i>object_names</i>	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_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_scenario

Removes a scenario in the multi-scenario analysis mode.

SYNTAX

```
string remove_scenario
scenario list
```

ARGUMENTS

```
scenario list
A list of unique strings used to identify each scenario.
```

DESCRIPTION

The **remove_scenario** command removes the specified scenario from memory. To determine the current scenarios that exist, execute the following:

```
report_multi_scenario_design -scenarios
1
```

EXAMPLES

In the following example, two scenarios named scen1 and scen2 are removed.

```
pt_shell> remove_scenario {scen1 scen2}
1
```

SEE ALSO

```
create_scenario(2)
report_multi_scenario_design(2)
```

remove_sense

Removes sense information defined on pins or cell timing arcs.

SYNTAX

```
string remove_sense
[-type type]
[-clocks clocks_object_list]
[-all]
object_list
```

Data Types

<i>clocks_object_list</i>	list
<i>object_list</i>	list

ARGUMENTS

-type *type*

Specifies whether the type of sense being removed refers to clock networks or data networks. The possible values for the *type* variable are: 'clock', 'data'. Note that 'clock' is assumed to be the default if -type option is not given.

-clocks *clocks_object_list*

Optionally specifies a list of clock objects to be associated with the given pin objects in the *object_list* variable. If the -clocks option is specified, only the unateness specified for that particular clock domain is removed. Otherwise, unateness information for all clocks passing through the given pin objects is removed. The -clocks option can only remove clock sense predefined by the **set_sense** -clock variable. It does not remove the default clock sense setting for this given pin.

-all

Removes all unateness information in current design.

object_list

Lists pins or cell timing arcs with predefined unateness to remove.

DESCRIPTION

Removes the unateness information which you predefined. To set the unateness information, use the **set_sense** command.

EXAMPLES

The following example removes positive unateness defined on a pin named **XOR/Z** with respect to clock **CLK1**, 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
```

The following 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

```
remove_setup_hold_pessimism_reduction
[-setup_cutoff]
[-hold_cutoff]
```

ARGUMENTS

```
-setup_cutoff
    Reset the setup_cutoff_slack value to negative INFINITY.

-hold_cutoff
    Reset the hold_cutoff_slack value to negative INFINITY.
```

DESCRIPTION

Remove the optimization constraints for setup-hold pessimism reduction (SHPR). The **setup_cutoff** command resets the *setup_cutoff* slack to negative INFINITY. The **hold_cutoff** command resets the *hold_cutoff* slack to negative INFINITY. A 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

```
int remove_si_aggressor_exclusion
[-rise]
[-fall]
[-all]
[nets]
```

Data Types

anets 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

```
set_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)
```

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

<i>rnets</i>	list
<i>inets</i>	list
<i>vnets</i>	list
<i>anets</i>	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 *-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_steady_state_resistance

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

<i>class_name</i>	string
<i>object_spec</i>	list
<i>attr_name</i>	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_waveform_integrity_analysis

Removes static or dynamic constraints for waveform integrity analysis.

SYNTAX

```
status remove_waveform_integrity_analysis
[-static]
[-dynamic]
[object_list]
```

Data Types

object_list list

ARGUMENTS

```
-static
    Removes the constraint for static waveform integrity analysis.

-dynamic
    Removes the constraint for dynamic waveform integrity analysis.

object_list
    Specifies a list of cell input pins.
```

DESCRIPTION

This command removes the constraint limit for waveform integrity analysis. The **-static** and **-dynamic** options specify which constraint is removed.

If the list of cell input pins are provided, the constraint limit on the specified cell inputs is removed. If there is no list of cell input pins, the design-level constraint value, if any, is removed.

EXAMPLES

The following example removes a static constraint on synchronous input pin "DFF/CLK".

```
pt_shell> remove_waveform_integrity_analysis -static 0.3 [get_pins DFF/CLK]
```

SEE ALSO

```
report_constraint(2)
report_waveform_integrity_analysis(2)
set_waveform_integrity_analysis(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_model
680

remove_wire_load_selection_group

Removes wire load selection_group from current design.

SYNTAX

```
string 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

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 infomration, 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

Change the name of a cell.

SYNTAX

```
int rename_cell
cell
new_name
```

Data Types

cell	list
new_name	string

ARGUMENTS

cell

Specifies a cell to be renamed. Only one cell can be specified.

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 **rename_cell** command is recorded for output with the **write_changes** command.

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

<i>design</i>	list
<i>new_name</i>	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_net

Change the name of a net.

SYNTAX

```
int rename_net
net
new_name
```

Data Types

<i>net</i>	list
<i>new_name</i>	string

ARGUMENTS

net list
Specifies a net to be renamed. Only one net can be specified.

new_name string
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.

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_design\(2\)](#)
[rename_cell\(2\)](#)
[write_changes\(2\)](#)

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

<i>SAIF / VCD</i>	string
<i>pipe</i>	string
<i>prefix</i>	string
<i>strip_path</i>	string
<i>pattern</i>	string
<i>file_name</i>	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.

```
f(CR
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
ributes

File File
Object Type Name

Design Design

Object Object
Type Name

Name

Att

```
-----
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.

```
f(CR
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

report_activity_file_check

692

```

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      Attr
butes
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.

```

f(CR
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    Togg  
le           Start        End       Rate  
-----  
-----  
TOP_level_of_my_design    1383        0        2000     0.00  
0192  
Second_level_of_my_design  1335        0        2000     0.00  
0199  
-----  
-----
```

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

<i>cell</i>	string
<i>lib_spec</i>	list
<i>delay_type</i>	string
<i>digits</i>	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 inout). 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 Library Cells	Slack
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 Library Cells	Slack (min:max)
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 Library Cells	Slack	Weighted max_delay/setup Cost	Total Design Cost
<hr/>			
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

```
string 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

<i>status_list</i>	list
<i>check_type_list</i>	list
<i>untested_reason_list</i>	list
<i>sort_method</i>	string
<i>digits</i>	integer
<i>pre_command_string</i>	string
<i>post_command_string</i>	string

ARGUMENTS

-status_details *status_list*

Specifies a space-separated 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**.

Note: 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-separated 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 allowed values are as follows:

- **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 method of sorting for the output of the detailed list. The allowed values are **slack**, **name**, and **check_type** (or its alias **check**). The default is **slack**, which sorts the output first by the slack (untested is treated as negative infinity slack), then by the pin name, then by the type of the check.

If you specify **name**, the output is sorted alphabetically by the name of the constrained pin, then by the slack, then by the check type.

If you specify **check_type** or **check**, the output is sorted alphabetically by the type of the check, then by the slack, then by the 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 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 command. Commands are grouped using the ";" character. The maximum size of a command is 1000 chars.

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.

You can see more information about the individual timing checks by using 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 HyperScale analysis flow, the following options are available for use at the top-level of the analysis:

```
[-full_design]  
[-list_all]  
[-instances cell_list]
```

The HyperScale-specific option **-full_design** produces an integrated report which shows the analysis coverage summary for the top level of the analysis combined with the coverage data from abstracted hierarchical instances. The option **-list_all** produces separate subreports for the top and block levels. To show separate reports for selected abstracted hierarchical instances, pass a list of hierarchical cells to the **-instances** option.

Note: If HyperScale-specific options are used with the **-status_details** option, the detailed information is shown for the top-level analysis only.

In the distributed analysis mode, only following option is supported at the Master:

```
[-full_design]
```

In the distributed analysis mode, if no option is provided with the command at the Master, then **-full_design** will be turned on by default. All the other options are supported with the command, if executed on remote slaves 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 Pin	Related Pin	Check 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 Pin	Related Pin	Check Type	Slack	Reason
Q2 ff4/TI	CP	out_setup setup	-2.30	untested no_paths

The following example shows results when command is run at the Master in distributed analysis mode:

```
pt_shell> report_analysis_coverage
```

```
Start of Master/Slave Task Processing
-----
Started : Task execution on 'partition3'
Started : Task execution on 'partition2'
Started : Task execution on 'partition1'
Started : Task execution on 'partition4'
Successful : Task execution on 'partition3'
Successful : Task execution on 'partition4'
Successful : Task execution on 'partition2'
Successful : Task execution on 'partition1'
-----
End of Master/Slave Task Processing
```

```
Start of Master/Slave Task Processing
-----
Started : Task execution on 'partition0'
Successful : Task execution on 'partition0'
-----
End of Master/Slave Task Processing
```

```
report_analysis_coverage -full_design;
*****
```

```
Report : analysis_coverage
-fuill_design
Design : bcm
```

```
report_analysis_coverage
```

Version: J-2014.12-alpha-DEV
Date : Wed Sep 24 11:13:39 2014

Type of Check	Total	Met	Violated	Untested
setup	48673	17481 (36%)	29899 (61%)	1293 (3%)
hold	48689	45630 (94%)	1766 (4%)	1293 (3%)
recovery	63	57 (90%)	6 (10%)	0 (0%)
removal	24	24 (100%)	0 (0%)	0 (0%)
min_period	46	46 (100%)	0 (0%)	0 (0%)
min_pulse_width	86034	85984 (100%)	10 (0%)	40 (0%)
out_setup	596	555 (93%)	29 (5%)	12 (2%)
out_hold	596	584 (98%)	0 (0%)	12 (2%)
All Checks	184721	150361 (81%)	31710 (17%)	2650 (1%)

1

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]
[-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.

-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
*****
```

	Total	Annotated	NOT 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
*****
```

	Total	Annotated	NOT 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
*****
```

	Total	Annotated	NOT 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
*****
```

	Total	Annotated	NOT 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	
<hr/>				
	38	38	0	

SEE ALSO

[read_sdf\(2\)](#)
[remove_annotated_delay\(2\)](#)
[reset_design\(2\)](#)
[set_annotated_delay\(2\)](#)

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]
[-constant_arcs]
[net_list]
```

Data Types

<i>num</i>	integer
<i>net_list</i>	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
Limits reporting to the specified maximum number of nets for the **-list_annotated** and **-list_not_annotated** options.

-list_annotated
Lists back-annotated nets. You cannot use this option together with the **-list_not_annotated** option.

-list_not_annotated
Lists nets that are not back-annotated. You cannot use this option together with the **-list_annotated** option.

-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
*****
```

Net Type	Total	Lumped	RC pi	RC network	Not 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
*****
```

Net Type	Total	Lumped	RC pi	RC network	Coupled network	Not 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
<hr/>						
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
<hr/>						
	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
*****
```

Net Type	Total	Lumped	RC pi	RC network	Coupled network	Not Annotated
<hr/>						
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
<hr/>						
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
<hr/>						
	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

```
int report_annotated_power
[-rails rail_list]
[-list_annotated]
```

ARGUMENTS

```
-rails rail_list
    Specifies a list of rails/power supply nets for which annotated power is
    reported.

-list_annotated
    Indicates to list powers that are annotated.
```

DESCRIPTION

This command provides a summary report of how many powers are annotated in the current design and, if the `-list_annotated` option is specified, a list of cells with power values annotated on them.

If the rail option is specified, only annotated power for the specified power supply nets (or rails in non-UPF mode) in the list will be reported. This feature is only valid when the variable `power_enable_multi_rail_analysis` is set to true.

EXAMPLES

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -int 1 -leak 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)

Cell type | Total | Annotated | NOT 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
*****
```

Cell type	Total	Annotated	NOT 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_aocvm

Reports information about advanced on-chip variation (OCV) derate tables and coefficients. Also displays details of path-based and graph-based advanced OCV calculation.

SYNTAX

```
status report_aocvm
[-early]
[-late]
[-rise]
[-fall]
[-clock]
[-data]
[-voltage voltage_value]
[-cell_delay]
[-net_delay]
[-list_annotated]
[-list_not_annotated]
[-coefficient]
[-nosplit]
[object_list]
```

Data Types

<i>voltage_value</i>	string
<i>object_list</i>	list

ARGUMENTS

-early

Displays only the early advanced OCV derate tables for the objects specified in the *object_list*. This argument can be used with the *-list_not_annotated* option.

-late

Displays only the late advanced OCV derate tables for the objects specified in the *object_list*. This argument can be used with the *-list_not_annotated* option.

-rise

Displays only the rise advanced OCV derate tables for the objects specified in the *object_list*. This argument can be used with the *-list_not_annotated* option.

-fall

Displays only the fall advanced OCV derate tables for the objects specified in the *object_list*. This argument can be used with the *-list_not_annotated* option.

-clock

Displays only the clock advanced OCV derate tables for the objects specified

in the *object_list*. This argument can be used with the **-list_not_annotated** option.

-data
 Displays only the data advanced OCV derate tables for the objects specified in the *object_list*. This argument can be used with the **-list_not_annotated** option.

-voltage voltage_value
 Displays only the voltage value-specific advanced OCV derate tables for the objects specified in the *object_list*.

-cell_delay
 Indicates that only cell delay advanced OCV derate tables are shown on the objects specified in the *object_list*.

-net_delay
 Indicates that only net delay advanced OCV derate tables are shown on the objects specified in the *object_list*.

-list_annotated
 Indicates that leaf cells and global nets that are annotated with advanced OCV derate tables are listed.

-list_not_annotated
 Indicates that leaf cells and global nets that are not annotated with advanced OCV derate tables are listed.

-coefficient
 Indicates that random advanced OCV coefficients are shown. 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.

-nosplit
 Do not split lines when columns overflow.

object_list
 Specifies either *timing_path* objects for which path-based advanced OCV metrics are to be reported; or *timing_arc* objects for which graph-based advanced OCV metrics are to be reported; or design objects on which advanced OCV derate tables and advanced OCV derate coefficients have been annotated.

DESCRIPTION

Displays the user-specified advanced OCV information. The type of information displayed by PrimeTime depends on the type of advanced OCV information annotated.

If the design has been annotated with advanced OCV derate tables using the **read_aocvm** command, the **report_aocvm** command outputs a summary showing the numbers of leaf cells and global nets annotated with advanced OCV derate tables. Lists of fully annotated, partially annotated, and not annotated leaf cells and global nets can be displayed using the **-list_annotated** and **-list_not_annotated** options. You can specify a collection of the design, library arcs, hierarchical cells, and nets in the *object_list* to display advanced OCV derate tables annotated only on those

objects. The early, late, rise, fall, clock, and data flags can be used with the list_not_annotated option to filter the list of netlist objects reporting only those missing the specified derating type(s). If a netlist object is specified and the advanced OCV information has been read from a file, the full path to the file is listed.

If the design has been annotated with advanced OCV derate coefficients using the **set_aocvm_coefficient** command, the **report_aocvm -coefficient** command displays these coefficients. You can specify a collection of library cells, library timing arcs, and cells in the *object_list* to display coefficients annotated only on those objects.

If a timing path is specified in the *object_list*, path-based metrics (distance, launch depth, and capture depth) for that path are displayed. Note that the depths shown in the report have been scaled by random advanced OCV coefficients, if they exist. Timing paths can be obtained using the **get_timing_paths** command. The path passed to **report_aocvm** has to be a path-based analysis path if a graph-based analysis path is specified an error is issued.

If a timing arc is specified in the *object_list*, graph-based metrics for that arc are displayed. Note that the depths shown in the report have been scaled by random advanced OCV coefficients, if they exist. Timing arcs can be obtained using the **get_timing_arcs** command. Graph-based timing arc metrics are only displayed when graph-based advanced OCV analysis has been enabled by setting the **timing_aocvm_enable_analysis** variable to **true**.

EXAMPLES

In the following example, the design has been annotated with advanced OCV derate factors using the **read_aocvm** command.

```
pt_shell> report_aocvm
```

```
*****
Report : aocvm
Design : my_design
*****
```

	Total	Fully annotated	Partially annotated	Not annotated
Leaf cells	6	0	4	2
Nets	7	0	0	7
	13	0	4	9

The following example displays path-based metrics for the timing path specified in the *object_list*, for a design that has been annotated with advanced OCV derates.

```
pt_shell> report_aocvm [get_timing_paths -pba path -path_type full_clock_expanded]
```

```
*****
Report : aocvm
object_list
```

```
Design : my_design
*****
```

```
Startpoint: ffL (rising edge-triggered flip-flop clocked by clk2)
Endpoint: ffC (rising edge-triggered flip-flop clocked by clk2)
Path Group: clk2
```

Path metrics	Cell	Net
Distance	575.85	666.41
Launch depth	2.69	3.07
Capture depth	1.00	1.09

The following example displays graph-based metrics and advanced OCV derate factors for the timing arc specified in the *object_list*. This report is only displayed when graph-based advanced OCV is enabled. The arc depth and distance displayed is the most pessimistic depth and distance for all possible paths through this arc.

```
pt_shell> report_aocvm [get_timing_arcs -from u2/A -to u2/Z]
```

```
*****
Report : aocvm
          object_list
Design : my_design
*****
```

```
From pin: u2/A
To pin:   u2/Z
Arc type: cell (data network)
```

AOCVM arc metrics	Launch
Distance	702.14
Depth	3.00

AOCVM arc derates	Launch
Early rise	0.8705
Early fall	0.8508
Late rise	1.1357
Late fall	1.1566

SEE ALSO

```
get_timing_paths(2)
read_aocvm(2)
remove_aocvm(2)
report_timing(2)
set_aocvm_coefficient(2)
timing_aocvm_enable_analysis(3)
```

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 \= b
The value must be greater than or equal to "b".

b \

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

<i>class_name</i>	string
<i>attribute_list</i>	list
<i>object_list</i>	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
--------	----------	--------	-------------	----------------	----------------

report_attribute

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.

```
-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

Cell	Reference	Bottleneck Cost
<hr/>		
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
baddrreg	out	15	0	16
bwordreg	out	15	0	16
caddrreg	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 case values and logic constant values throughout the netlist.

SYNTAX

```
string report_case_analysis
[-all]
[-nosplit]
[-to]
[-from]
[pin_or_port_list]
```

Data Types

pin_or_port_list list

ARGUMENTS

-all

Reports the pins upon which you 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. 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.

-to

Invokes PrimeTime GCA and 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

Invokes PrimeTime GCA and 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.

```
pin_or_port_list
    Specifies a list of pins or ports.
```

DESCRIPTION

This command reports case analysis entries 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. The list does not report where the logic constant values are propagated. If you specify *pin_or_port_list*, PrimeTime GCA shows the propagated case values on the specified pins or ports. If you specify the **-from** or **-to** option with the *pin_or_port_list*, PrimeTime GCA reports the case fanout and fanin tracing, which shows how case values propagate from or to the specified pins or ports.

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
```

report_case_analysis

734

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

SEE ALSO

```
remove_case_analysis(2)  
set_case_analysis(2)
```

report_cell

Reports cell information.

SYNTAX

```
string report_cell
[-connections [-verbose]]
[-significant_digits digits]
[-nosplit]
[cell_names]
```

Data Types

<i>digits</i>	integer
<i>cell_names</i>	list

ARGUMENTS

-connections

Displays the pins and the nets to which they connect.

-verbose

Displays verbose 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 in conjunction with the **-connections** option.

-nosplit

Does not split lines if column overflows.

-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. 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_names

Lists cells to report. If you do not specify this option, all cells in the current instance are listed.

DESCRIPTION

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.

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
Design : middle
Version: Y-2006.06
Date   : Wed Mar  8 03:18:34 2006
*****
```

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	

1

The following example gives a verbose report of the cell connections.

```
pt_shell> report_cell -verbose -connections
```

```
*****
Report : cell
        -connections
        -verbose
Design : middle
```

Version: Y-2006.06

Date : Wed Mar 8 03:18:44 2006

Connections for cell 'il':

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 out_2 i1/low/n1/B i1/low2/n1/B	Input Pin (NR4) Output Port Input Pin (NR4) Input Pin (NR4)
QN	NET2QNX	i2/low3/n1/B i2/low/n1/B i2/low2/n1/B	Input Pin (NR4) Input Pin (NR4) Input Pin (NR4)

1

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

<i>pattern_list</i>	list
<i>attribute_name</i>	string
<i>power_attribute_name</i>	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 obsoleted in future release and its feature is covered by the **report_eco_library_cells** command.

-show_others
Shows library cells that do not match the specified pattern. This option will be obsoleted in future release and its feature is covered by the **report_eco_library_cells** command.

-power_attribute *power_attribute_name*
Specifies an attribute for prioritizing library cells for downsizing. When you specify this option, the summary of power attribute value for each cell group is shown 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 specified. If the **-pattern_priority** option is not specified, 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
<hr/>			
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%)
<hr/>			
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

```
string report_clock
[-attributes]
[-skew]
[-groups]
[-map]
[-map_of instance_list]
[-cell hierarchical_cell_list]
[-include internal | virtual]
[-nosplit]
[clock_names]
```

Data Types

<i>instance_list</i>	list
<i>clock_names</i>	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

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

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.

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, user can use command **report_clock -map**. User specified mapping from **set_clock_map** is annotated with a '*' in the report.

To obtain a list of all clocks in the design, use the **all_clocks** command.

report_clock

744

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  
...  
*****
```

Object	Rise Delay	Fall Delay	Hold Uncertainty	Setup 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

Object	Rise Transition	Fall 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
```

```
...
```

```
*****
```

Object	Min Condition Source Latency				Max Condition Source Latency				Re
	Early_r	Early_f	Late_r	Late_f	Early_r	Early_f	Late_r	Late_f	
l_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
```

```
report_clock
```

```
746
```

```

...
*****
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
               B
               X          v
               n1_t4_CLK    v
               n_t2c4_CLK   v
               C
core2          X
               n1_t4_CLK    v
               n_t2c4_CLK   v
               C


```

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)

```

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

<i>clock_list</i>	list
<i>sort_method</i>	string
<i>object_list</i>	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

`report_clock_gate_savings`

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 Distribution	Number of Registers	% of 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 Registers	Clock Gate Stage	# of Clock Gates	# of Registers	Clock Gating Efficiency
CLK	12	0	0	0	NA

1	2	8	30.61%				
2	1	4	54.52%				
<hr/>							
<hr/>							
Clock Gate Structure Details							
<hr/>							
Clock Receiver	Clock Gate	Gating Element	# of Fan	# of Clock	# of Registers	Toggle Savings	Clock Gating
Clock Stage			Out	Gates		(%)	Efficiency
Gates							
<hr/>							
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/latch	
<hr/>							

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 Clock Gates	Toggle Savings (%)
<hr/>		
TOP	225	67%
/A	125	50%
//B	30	12%
/C	100	80%
<hr/>		

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 Toggle Rate	Local Clock Toggle Rate	Q Toggle Rate	Q/Clk Toggle Ratio	Register Gating 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 Registers	Q/Clk Toggle Ratio	Avg Register Gating 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

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
...
*****
```

Cell	Enable	Clock	Rise		Fall		High/Low	Attr
			Setup	Hold	Setup	Hold		
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]
[-sort_by sort_attr]
[-clock_crossing]
[-include_clock_gating]
[-nosplit]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
```

Data Types

<i>report_type</i>	string
<i>clock_list</i>	list
<i>from_clock_list</i>	list
<i>to_clock_list</i>	list
<i>from_list</i>	list
<i>to_list</i>	list
<i>worst_entries</i>	int
<i>lower_limit</i>	float
<i>upper_limit</i>	float
<i>slack_upper_limit</i>	float
<i>digits</i>	int
<i>sort_attr</i>	string
<i>pre_command_string</i>	string
<i>post_command_string</i>	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.

-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
f2_1/CP            2.01     0.11     4.21      rp-+
                                         fp-+
```

13_2/G	4.10			rpi-
f1_2/CP	1.00	0.22	3.32	rpi--
12_2/G	4.11			rp-
f1_2/CP	1.00	0.12	3.23	rpi--
f2_2/CP	6.11			rp-+
13_3/G	3.01	0.11	3.21	rp-
11_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

Clock Pin	Trans	--- Latency ---			
		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
<hr/>			
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
<hr/>			
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
<hr/>			
startpoint clock latency			6.11
endpoint clock latency			-2.01
<hr/>			
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
<hr/>			
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-+
12_2/G	4.01			rp-	
f1_2/CP	1.10	-0.21	2.70	-6.64	rpi--
11_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
f2_1/CP          0.13533    0.02825  0.26688  0.34732   rp-+
12_1/CP          0.33036
12_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_clock_timing
768

report_constraint

Displays constraint-related information about a design.

SYNTAX

```
int report_constraint
[-all_violators]
[-verbose]
[-path_type slack_only | end]
[-max_delay]
[-min_delay]
[-max_capacitance]
[-min_capacitance]
[-max_transition]
[-min_transition]
[-max_fanout]
[-min_fanout]
[-min_pulse_width]
[-min_period]
[-pulse_clock_min_width]
[-pulse_clock_max_width]
[-pulse_clock_min_transition]
[-pulse_clock_max_transition]
[-recovery]
[-removal]
[-max_skew]
[-clock_gating_setup]
[-clock_gating_hold]
[-clock_separation]
[-connection_class]
[-boundary_check]
[-boundary_check_include boundary_check_group_list]
[-ignore_register_feedback feedback_slack_cutoff]
[-significant_digits digits]
[-nosplit]
[-format style]
[-output file_name]
[-default_scenario_name scen_name]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-waveform_integrity static | dynamic]
[object_list]
```

Data Types

<i>boundary_check_group_list</i>	list
<i>feedback_slack_cutoff</i>	float
<i>digits</i>	int
<i>style</i>	string
<i>file_name</i>	string
<i>scen_name</i>	string
<i>pre_command_string</i>	string
<i>post_command_string</i>	string

object_list list

ARGUMENTS

-all_violators

Displays a summary that shows the worst violation per endpoint of each violated design rule constraint in the current design. The **-verbose** option provides detailed information about each constraint violation. Multiple violations for a given constraint are listed from the greatest to the least violator.

-verbose

Indicates that more detail is to be shown about constraint calculations.

-path_type slack_only | end

Specifies the format for the path report. Allowed values are **slack_only** (the default) and **end**. This option has an effect only if the **-verbose** option is not used. If you specify **slack_only**, the report displays only endpoint slacks. If you specify **end**, the report has a column format that shows one line for each path, with only the endpoint path total, required time, and slack.

-max_delay

Indicates that only maximum delay and setup information is to be displayed. In the default report, the worst violation per clock group is displayed. The default constraint report displays all timing and design rule constraints.

-min_delay

Indicates that only minimum delay and hold information is to be displayed. In the default report, the sum of all violations per clock group is displayed. The default constraint report displays all timing and design rule constraints.

-max_capacitance

Indicates that only maximum capacitance constraint information is to be displayed. The **-max_capacitance** option is a design rule used to limit 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 violator, use the **-verbose** option in addition to the **-max_capacitance** option. To see details about all max_capacitance violations, use the **-all_violators** and **-verbose** options in addition to the **-max_capacitance** option. Note that max_capacitance constraint is not checked for load pin. The default constraint report displays all timing and design rule constraints.

-min_capacitance

Indicates that only minimum capacitance constraint information is to be displayed. The **-min_capacitance** option is a design rule used to limit total capacitance on a net. The default constraint report displays all timing and design rule constraints.

-max_transition

Indicates that only maximum transition constraint information is to be displayed. **-max_transition** is a design rule used to limit transition time on

a ports and pins. The default constraint report displays all timing and design rule constraints. If the library uses the cmos2 delay model, maximum edge rate information is shown instead.

-min_transition

Indicates that only minimum transition constraint information is to be displayed. **-min_transition** is a design rule used to set a minimum transition time on a ports and pins. The default constraint report displays all timing and design rule constraints. If the library uses the cmos2 delay model, maximum edge rate information is shown instead.

-max_fanout

Indicates that only maximum fanout constraint information is to be displayed. The **-max_fanout** option is a design rule used to limit fanout_load on a net. The default constraint report displays all timing and design rule constraints.

-min_fanout

Indicates that only minimum fanout constraint information is to be displayed. The **-min_fanout** option is a design rule used to set a minimum fanout_load on a net. The default constraint report displays all timing and design rule constraints.

-min_pulse_width

Indicates that only minimum pulse width constraint information is to be displayed. The **-min_pulse_width** option is a design rule used to set a minimum pulse width high or low at a clock pin or at pins or ports in the clock network. The default constraint report displays all timing and design rule constraints.

-min_period

Indicates that only minimum period constraint information is to be displayed. The **-min_period** option is a design rule used to set a minimum period on a clock signal. The default constraint report displays all timing and design rule constraints.

-pulse_clock_min_width

Indicates that only pulse clock minimum width constraint information is to be displayed.

-pulse_clock_max_width

Indicates that only pulse clock maximum width constraint information is to be displayed.

-pulse_clock_min_transition

Indicates that only pulse clock minimum transition constraint information is to be displayed.

-pulse_clock_max_transition

Indicates that only pulse clock maximum transition constraint information is to be displayed.

-recovery

Indicates that only recovery constraint information is to be displayed. 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. The default constraint report displays all timing and design rule constraints.

-removal

Indicates that only removal constraint information is to be displayed. 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. The default constraint report displays all timing and design rule constraints.

-max_skew

Indicates that only maximum skew constraint information is to be displayed. The **-max_skew** option is a timing constraint that checks the maximum separation time allowed between two clock signals. The default constraint report displays all timing and design rule constraint.

-clock_gating_setup

Indicates that only clock gating setup constraint information is to be displayed. 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. The default constraint report displays all timing and design rule constraints.

-clock_gating_hold

Indicates that only clock gating hold constraint information is to be displayed. 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. The default constraint report displays all timing and design rule constraints.

-clock_separation

Indicates that only clock separation constraint information is to be displayed. The **-clock_separation** option is a timing constraint that checks the minimum separation time allowed between two clock signals. The default constraint report displays all timing and design rule constraint.

-connection_class

Indicates to display only 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.

-ignore_register_feedback feedback_slack_cutoff

Indicates to ignore 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.

-boundary_check

Displays only HyperScale boundary constraint information. Boundary constraints are only displayed during top-level HyperScale runs when **hyperscale_enable_analysis** is set to **true**, and there are boundary constraint violations between chip and instance. If there is an asterisk beside the constraint name in the summary output of the report, then the tool automatically generates updated boundary constraints to resolve these violation.

-boundary_check_include boundary_check_group_list

Controls the subset of boundary check constraints that are reported by **report_constraint**.

-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

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 to facilitate parsing of information from the report output.

-format style

Choose reporting format. Allowable value for *style* is *csv*.

-output file_name

Specifies the output file name for CSV report.

-default_scenario_name scen_name

This option allows a scenario name to be specified in the CSV report. This option is valid only with **-format csv** and **-output** options. Note that, in DMSA slaves, this option will override the implicit slave scenario name by the provided scenario name, in the CSV reports.

-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.

```
-waveform_integrity static | dynamic
The static option is available only if the advanced waveform propagation is
enabled. The dynamic option is available only if the signal integrity
analysis is enabled. The constraints used for waveform integrity have to be
set using set_waveform_integrity_analysis command.
```

object_list

Specifies a list of pins, ports and 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*, *max_capacitance*, or both options, and 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

The **report_constraint** command displays the following information for the constraints on the current design: whether the constraint is violated or met, by how much the constraint value is violated or met, and the design object that is the worst violator.

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.

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.

The **report_min_pulse_width** command displays information similar to that displayed by the **report_constraint -min_pulse_width** command.

The **report_pulse_clock_min_width** command displays information similar to that displayed by the **report_constraint -pulse_clock_min_width** command.

The **report_pulse_clock_max_width** command displays information similar to that displayed by the **report_constraint -pulse_clock_max_width** command.

The **report_pulse_clock_min_transition** command displays information similar to that displayed by the **report_constraint -pulse_clock_min_transition** command.

The **report_pulse_clock_max_transition** command displays information similar to that displayed by the **report_constraint -pulse_clock_max_transition** command.

To enable pulse clock constraint checking, set the **timing_enable_pulse_clock_constraints** variable to *true*. This variable is set to *false* by default. The *object_list* can be optionally specified only with the *max_transition* and *max_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

In the HyperScale analysis flow, you can use the following options:

```
[-boundary_check]
[-boundary_check_include boundary_checks]
```

You can use the **-boundary_check** and **-boundary_check_include** HyperScale options at the top-level HyperScale analysis. The **-boundary_check** option reports only the HyperScale boundary constraint violations. The **-boundary_check_include** option reports the specified boundary checks; you can use the following values for *boundary_checks*:

- **all** - Reports all violating boundary checks
- **auto_fixable** - Reports only violated checks that can be fixed automatically in subsequent iterations of HyperScale analysis
- **clock_mapping** - Reports failures to map top-level clocks to block-level clocks
- **clock_attributes** - Reports attribute mismatches between top-level clocks and block-level clocks)

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
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
<hr/>		
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
<hr/>		
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
<hr/>		
data required time		9.10
data arrival time		-5.22
<hr/>		
slack (MET)		3.88

Design: counter

max_area	30.00
- Current Area	78.00
<hr/>	
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
```

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 slack_only
  -max_transition
...
*****
```

max_transition

Pin	Required Transition	Actual 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

Pin	Required pulse width	Actual 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

report_constraint

778

Pin	Required pulse width	Actual 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 slack_only
    -max_skew
...
*****
```

Pin	Required Skew	Actual Skew	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	default
U18/B		default
U20/B		

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)`
`report_default_significant_digits(3)`

report_context

Reports the characterized timing context information.

SYNTAX

```
string report_context
[-timing]
[-environment]
[-design_rules]
[-constant_inputs]
[-nosplit]
cell_list
```

Data Types

cell_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.

cell_list
Lists cells for which to report the context.

DESCRIPTION

Reports the timing context captured using the **characterize_context** command for the specified list of instances.

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
```

SEE ALSO

```
characterize_context(2)  
write_context(2)  
remove_context(2)
```

report_crpr

Reports the clock reconvergence pessimism (CRP) calculated between specified register clock pins or ports.

SYNTAX

```
int 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

<i>from_latch_clock_pin</i>	string
<i>to_latch_clock_pin</i>	string
<i>from_clock</i>	string
<i>to_clock</i>	string
<i>digits</i>	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 both 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
<hr/>			
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
<hr/>			
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_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

<i>from_pin</i>	string
<i>to_pin</i>	string
<i>objects</i>	collection
<i>value</i>	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 Thresholds:	Rise Delay	Rise Slew	Fall Delay	Fall Slew	Slew Derate	Rail 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 Thresholds:	Rise Delay	Rise Slew	Fall Delay	Fall Slew	Slew Derate	Rail Voltage	Rail Temp.
<hr/>							
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.
<hr/>							
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
  -----
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

Displays attributes on the **current_design**.

SYNTAX

```
status report_design
[-nosplit]
```

ARGUMENTS

-nosplit

Prevents line splitting and facilitates scripting 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

Lists information about the attributes on the **current_design** command.

EXAMPLES

The following is an example of 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 pin has any mismatched issues, the **is_design_mismatch** attribute is set on its connected net and cell and itself. If it is a bus, all other bits of the bus are also marked with this 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 example generates a report when the pin u1/Z and u2/Z 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_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_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 and 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

p : The propagation of constant values

u : Arcs disabled by using the **set_disable_timing** command.

EXAMPLES

The following example shows that the timing arc of a cell named *U1/U2* 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
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
            p  propagated constant
            u  user-defined
```

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 cell *ff4* 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
            p  propagated constant
            u  user-defined
```

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		

report_disable_timing

```
c      D = 0  
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  
          p  propagated constant  
          u  user-defined
```

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

```
int 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

<i>lib_cell</i>	string
<i>from_pin</i>	string
<i>to_pin</i>	string
<i>rise_slew</i>	float
<i>fall_slew</i>	float
<i>capacitance</i>	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 IO] \
    -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_eco_library_cells

Reports alternative library cells considered for PrimeTime ECO commands.

SYNTAX

```
string report_eco_library_cells
[-pattern_priority pattern_list]
[-attribute attribute_name]
[-show_others]
[-current_library]
[-nosplit]
[-power_attribute power_attribute_name]
```

Data Types

<i>pattern_list</i>	list
<i>attribute_name</i>	string
<i>power_attribute_name</i>	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 to match patterns specified by the **-pattern_priority** option. When you specify this option, PrimeTime uses the value of the attribute instead of library cell names for pattern matching.
- show_others**
Shows library cells that do not match the specified pattern. This option must be used together with the **-pattern_priority** option.
- current_library**
Shows alternative library cells within the same library.
- nosplit**
Prevents line splitting when column overflows to facilitate writing script to extract information from the report output.
- power_attribute *power_attribute_name***
Specifies an attribute for prioritizing library cells for downsizing. If you specify this option, the power attribute values of library cells are shown in a separate column.

DESCRIPTION

This command reports the alternative library cells of all cells in a design. This command must be used by slaves for multi-scenario analysis flow within PrimeTime.

EXAMPLES

The following example reports alternative library cells:

```
pt_shell> report_eco_library_cells
```

```
*****
Report : eco_library_cells
Design : design
...
*****
```

Alternative library cells:

Attributes:

 u - dont_use or pt_dont_use
 d - dont_touch

Group	Library_Cell	Area	Attributes
[0]	LIBRARY/BUFX1_LVT	2.12	
	LIBRARY/BUFX2_LVT	2.64	
	LIBRARY/BUFX4_LVT	4.76	d
	LIBRARY/BUFX8_LVT	7.41	u d
[1]	LIBRARY/DFFPQX1_LVT	9.52	
	LIBRARY/DFFPQX2_LVT	9.52	
	LIBRARY/DFFPQX4_LVT	12.70	

The following example reports alternative library cells with patterns HVT and LVT:

```
pt_shell> fBreport_eco_library_cells -pattern_priority {HVT LVT}
```

```
*****
Report : eco_library_cells
      -pattern_priority { HVT LVT }
Design : design
...
*****
```

Alternative library cells:

Attributes:

 u - dont_use or pt_dont_use
 d - dont_touch

Group	Library_Cell	Area	Attributes
[0]	LIBRARY/BUFX1_HVT	2.12	
	LIBRARY/BUFX1_LVT	2.12	
[1]	LIBRARY/BUFX2_HVT	2.64	
	LIBRARY/BUFX2_LVT	2.64	
[2]	LIBRARY/DFFX1_HVT	9.52	u
	LIBRARY/DFFX1_LVT	9.52	

The following example reports alternative library cells with power attribute:

report_eco_library_cells

```

pt_shell> report_eco_library_cells -power_attribute power_attr
*****
Report : eco_library_cells
      -power_attribute power_attr
Design : design
...
*****

```

Alternative library cells:

Attributes:

- u - dont_use or pt_dont_use
- d - dont_touch

Group	Library_Cell	Area	Power_Attr	Attributes
[0]	LIBRARY_H/INV_X0_HVT	1.27	13773.83	
	LIBRARY_H/INV_X1_HVT	1.27	25491.85	
	LIBRARY_R/INV_X0_RVT	1.27	26041.65	
	LIBRARY_R/INV_X1_RVT	1.27	48138.68	
	LIBRARY_H/INV_X2_HVT	1.52	53638.71	
	LIBRARY_R/INV_X2_RVT	1.52	101260.30	
	LIBRARY_H/INV_X4_HVT	2.03	117975.20	
	LIBRARY_R/INV_X4_RVT	2.03	222585.09	u

SEE ALSO

[fix_eco_leakage\(2\)](#)
[fix_eco_power\(2\)](#)
[eco_alternative_cell_attribute_restrictions\(3\)](#)

report_eco_options

Reports options specified by the **set_eco_options** command.

SYNTAX

```
string report_eco_options
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_eco_options** command reports all the ECO options specified by the **set_eco_options** command.

Distributed Multi-Scenario Analysis

To execute the **report_eco_options** command in all scenarios, use the **remote_execute** command. For example:

```
remote_execute -verbose {  
    report_eco_options  
}
```

Note: You cannot execute the **report_eco_options** command in the master process.

EXAMPLES

The following example reports the ECO options previously specified by the **set_eco_options** command.

```
pt_shell> report_eco_options  
*****  
Report : eco_options  
Design : DESIGN  
...  
*****  
  
LEF file  
-----  
top.lef  
block1.lef  
block2.lef  
block3.lef  
  
DEF file          Physical constraint file  
-----  
top.def          top_va.tcl  
block1.def       block1_va.tcl
```

block2.def	block2_va.tcl
block3.def	block3_va.tcl
Log file	Log format
-----	-----

SEE ALSO

`reset_eco_options(2)`
`set_eco_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

<i>from_object</i>	list
<i>rise_from_object</i>	list
<i>fall_from_object</i>	list
<i>to_object</i>	list
<i>rise_to_object</i>	list
<i>fall_to_object</i>	list
<i>arc_type</i>	list
<i>path_type_list</i>	list
<i>levels</i>	integer
<i>er_file_name</i>	string
<i>nr_file_name</i>	string
<i>digits</i>	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

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list

ARGUMENTS

Note: Options marked with asterisks (*) in the SYNTAX can be used more than one time in the same command.

-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 **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.

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.

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
<hr/>				
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
<hr/>				
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
<hr/>				
QA	*	max=10	-	f
*	A	cycles=2	-	t
*	B	FALSE	FALSE	t

Object	Type	Attributes
<hr/>		
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
<hr/>					
{ 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_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

<i>digits</i>	integer
<i>port_pin_list</i>	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.

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.

report_global_slack

```
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)`

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 {[inter_clock] [non_violated] [per_clock_violations] [scenario_details]}}
[-format {[narrow | wide] [csv]}}
[-output file_name]
[-separate_non_standard_groups]
[-separate_all_groups]
[-significant_digits digits]
[-pba_mode none | path | exhaustive]
```

Data Types

group_list	list
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 {[inter_clock] [non_violated] [per_clock_violations] [scenario_details]}
Specifies additional reporting options. The following values are allowed:

- **inter_clock** - Shows 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_clockViolations** - 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 {[narrow | wide] [csv]}
Specifies the reporting format. 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.

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 HyperScale analysis flow, the following options are available for use at the top-level of the analysis:

```
[-full_design]
[-list_all]
[-instances cell_list]
[-pba_mode none | path | exhaustive]
```

The HyperScale-specific **-full_design** option produces an integrated report which shows summary of timing violations at the top-level STA combined with violations from the analysis of abstracted hierarchical instances. The **-list_all** option produces separate subreports for the top and block levels. To show separate reports for selected abstracted hierarchical instances, pass a list of hierarchical cells to the **-instances** option. The **-pba_mode** options enables the path-based analysis mode.

In the distributed analysis mode, only following option is supported at the Master:

[-full_design]

In the distributed analysis mode, if no option is provided with the command at the Master, then **-full_design** will be turned on by default. All the other options are supported with the command, if executed on remote slaves 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_clock per_clock}
*****
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 |  W
  NS      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 |  W
  NS      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,i
n2reg_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.

The following example shows results when command is run at the Master in the distributed analysis mode:

```
pt_shell> report_global_timing
```

```
Start of Master/Slave Task Processing
```

```
-----
Started    : Task execution on 'partition2'
Started    : Task execution on 'partition4'
Started    : Task execution on 'partition3'
Started    : Task execution on 'partition1'
Successful : Task execution on 'partition4'
Successful : Task execution on 'partition3'
Successful : Task execution on 'partition1'
Successful : Task execution on 'partition2'
-----
```

```
End of Master/Slave Task Processing
```

```
Start of Master/Slave Task Processing
```

```
-----
Started    : Task execution on 'partition0'
Successful : Task execution on 'partition0'
-----
```

```
End of Master/Slave Task Processing
```

```
report_global_timing -full_design;
*****
Report : global_timing
-format { narrow }
-full_design
Design : bcm
Version: J-2014.12
Date   : Wed Sep 17 08:18:51 2014
*****
```

```
Setup violations
```

```
-----
          Total    reg->reg    in->reg    reg->out    in->out
-----
WNS      -35.17     -35.17     -10.24     -10.85      0.00
TNS     -126110.92 -125794.02   -282.39     -34.51      0.00
NUM      30095      29678        388        29          0
-----
```

report_global_timing

Hold violations

	Total	reg->reg	in->reg	reg->out	in->out
WNS	-1.49	-1.49	0.00	0.00	0.00
TNS	-395.89	-395.89	0.00	0.00	0.00
NUM	1766	1766	0	0	0

1

SEE ALSO

`report_analysis_coverage(2)`
`report_timing(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

```
int report_host_usage
[-verbose]
[-nosplit]
[host_option_names]
```

Data Types

host_option_names list

ARGUMENTS

-verbose

Indicates you want to generate a more detailed report.

-nosplit

Do not split lines when columns overflow.

host_option_names

A list of names of the host options to report. The host options must have been defined using the **set_hosts_options** command. If no host option names are specified, then all host options are reported.

DESCRIPTION

The **report_host_usage** command generates a report showing the values specified for each host options set. The command also reports peak memory and CPU usage for the local process and all online distributed processes.

Calling the **report_host_usage** command before any hosts have been started reports the following basic information.

- Options Name

The name of the host options set.

- Host

The host on which the processes is launched. If the processes are to be launched on a managed compute resource, the >>farm<< is shown.

- Num Processes

The number of processes to be launched.

If you specify the `-verbose` option, the following additional information is reported.

- Action

The actions that can be performed for the host options.

SUBMIT : Command to submit a process for execution.
TERMINATE : Command to terminate a job that has not started.

- Command Options

The user-defined command to perform the associated action.

- Peak Memory (MB)

The peak memory usage of the current pt_shell process in megabytes.

- CPU Times

The total CPU time usage (in seconds) associated with the current pt_shell process.

After the hosts have been started using the **start_hosts** command, calling the **report_host_usage** command additionally shows the current "real time" details of the processes launched by default.

- #

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. Otherwise, it is the ID of the process that launched the slave PrimeTime process. 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.

- Process ID

This is 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

This is 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 and slave processes are different versions of PrimeTime and the slave process is not usable.
SHUTTING_DOWN	: The process is shutting down.

The command also reports the host options cores usage limits. The first entry in this table always captures the limits on the current or 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 as well as licensing restrictions.

If you have specified the `-verbose` option, the following columns are prepended:

- User

This captures the **max_cores** that you set in the corresponding **set_host_options** command.

- Hardware

This is the number of logical cores on the target hardware.

- Licensed

This accounts for what the current license count dictates in terms of cores usage. Note that the PrimeTime licensing manager always acquires the minimum license count to satisfy the user/hardware cores usage limit. However, since a single license is virtually equivalent to multiple cores, the possibility exists that the licensed cores limit is greater than the effective cores limit. While these residual cores can be used by the local or any remote process reported, their number is always reported in the row corresponding to the local process (** local process **). In a distributed multi-core session, licenses are shared between the master process and its slave processes. In this the number of licensed cores would include cores in both categories.

- Memory (MB)

This reports memory usage of the local and all distributed pt_shell processes currently online.

- CPU Times

This shows total CPU time usage associated with the local process.

In the multi-scenario analysis mode, the report also displays the CPU time for all distributed pt_shell processes currently online.

- Elapsed Times

This shows total elapsed time associated with the local process.

In the multi-scenario analysis mode, the report also displays the elapsed time for all distributed pt_shell processes currently online.

EXAMPLES

In the following example, the master process is running on the host named ptopt021 using a 64-bit binary and specifies five different sets of host options.

The first host options, named "my_opts1", specifies one process with the same architecture as the master on the same host as the master and does not specify an upper limit on the number of cores to use; therefore, the default applies.

The second host options, named "my_opts2", specifies two processes on the same host as the master and explicitly mentions the name of the master's host and specifies to use a maximum of two cores per process.

The third host options, named "my_opts3", specifies four processes (with the same architecture as the master) on the host named ptopt010 using "rsh" to connect to ptopt010 and specifying to use a maximum of three cores per process.

The fourth host options named "my_opts4", specifies five processes (with the same architecture as the master) on an LSF farm, requesting 500MB of memory and two slots per compute server and specifying to use a maximum of two cores per process. 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. A command is also provided for the termination of jobs that do not come online.

The fifth host options, named "my_opts5", specifies two processes (with the same architecture as the master) on a Grid farm requiring the jobs to be launched using the project named "bnormal" and does not specify an upper limit on the number of cores to use; therefore, the default applies. A command is also provided for the termination of jobs that do not come online. Note: The **-b y** option is required when accessing the grid.

Note: The host name of the host for my_opts1 is inferred as localhost (the master). The host name for my_opts1 and my_opts2 are surrounded by " ** ", indicating that the processes are launched directly on the host on which the master process is running. The host name for all managed resources is ">>farm<<".

Call the **set_host_options** command to specify the host options.

```
pt_shell> report_host_usage
*****
Report : host_usage
```

```
Version: E-2010.12
Date   : Fri Oct  1 15:32:12 2010
*****
```

Usage limits (cores)

Options Name	#	Effective
(local process)	-	4
Total		4

Memory

Options Name	#	Memory (MB)
(local process)	-	26.98

Performance

Options Name	#	CPU Time (s)	Elapsed Time (s)
(local process)	-	3	15

```
1
pt_shell> set_host_options -name my_opts1 -num_processes 1
1
pt_shell> set_host_options -name my_opts2 -num_processes 1 \
           ptopt018
1
pt_shell> set_host_options -name my_opts3 -num_processes 1 \
           -submit_command "/usr/bin/rsh -n" ptopt016
1
pt_shell> set_host_options -name my_opts4 -num_processes 5 \
           -submit_command "/lsf/bin/bsub -n 2 -R rusage\[mem=500\] \
           -R span\[ptile=2\]" -terminate_command "/lsf/bin/bkill"
1
pt_shell> set_host_options -name my_opts5 -num_processes 2 \
           -submit_command "/grid/bin/qsub -b y -P bnormal" \
           -terminate_command "/grid/bin/qdel"
1
```

To report the host options created by the **set_host_options** command, call the **report_host_usage** command with the **-verbose** option before starting the hosts.

Note: The submit command for my_opts1 and my_opts2 are empty because the processes are directly launched on the masters host.

```
pt_shell> report_host_usage -verbose
*****
Report : host_usage
         -verbose
Version: E-2010.12
Date   : Fri Oct  1 16:09:39 2010
```

```
*****
```

Options Name	Host Name	32Bit	Num Processes
<hr/>			
my_opts1	**localhost**	N	1
my_opts2	**localhost**	N	1
my_opts3	ptopt016	N	1
my_opts4	>>farm<<	N	5
my_opts5	>>farm<<	N	2
Options Name	Host Name	Action	Command
<hr/>			
my_opts1	**localhost**	SUBMIT	<execute directly on masters host>
my_opts2	**localhost**	SUBMIT	<execute directly on masters host>
my_opts3	ptopt016	SUBMIT	/usr/bin/rsh -n
my_opts4	>>farm<<	SUBMIT	/lsf/bin/bsub -n 2 -R rusage[mem=500] -R span[ptile=2]
my_opts5	>>farm<<	TERMINATE	/lsf/bin/bkill
		SUBMIT	/grid/bin/qsub -b y -P bnormal
		TERMINATE	/grid/bin/qdel

Usage limits (cores)

Options Name	#	User-set	Hardware	Licensed	Effective
<hr/>					
(local process)	-	none	4	4	4
my_opts1	1	-	-	-	-
my_opts2	1	-	-	-	-
my_opts3	1	-	-	-	-
my_opts4	1	-	-	-	-
my_opts5	1	-	-	-	-
<hr/>					
Total			4	4	4

Memory

Options Name	#	Memory (MB)
<hr/>		
(local process)	-	27.04

Performance

Options Name	#	CPU Time (s)	Elapsed Time (s)
<hr/>			
(local process)	-	3	15

1

To start the slave processes, call the **start_hosts** command. Notice that each instance launched is assigned a number, such as 1] Launched ...

```
pt_shell> start_hosts
1] Launched : ...
  Status   : Forking successful
  Stdout   : Process group is (945)
  Stderr   : **<<EMPTY>>**

2] Launched : ...
  Status   : Forking successful
  Stdout   : Process group is (980)
  Stderr   : **<<EMPTY>>**

3] Launched : ...
  Status   : Forking successful
  Stdout   : Process group is (20464)
  Stderr   : **<<EMPTY>>**

4] Launched : ...
  Status   : Forking successful
  Stdout   : Job <317496> is submitted to default queue <normal>.
  Stderr   : **<<EMPTY>>**

5] Launched : ...
  Status   : Forking successful
  Stdout   : Job <317497> is submitted to default queue <normal>.
  Stderr   : **<<EMPTY>>**

6] Launched : ...
  Status   : Forking successful
  Stdout   : Job <317498> is submitted to default queue <normal>.
  Stderr   : **<<EMPTY>>**

7] Launched : ...
  Status   : Forking successful
  Stdout   : Job <317499> is submitted to default queue <normal>.
  Stderr   : **<<EMPTY>>**

8] Launched : ...
  Status   : Forking successful
  Stdout   : Job <317500> is submitted to default queue <normal>.
  Stderr   : **<<EMPTY>>**

9] Launched : ...
  Status   : Forking successful
  Stdout   : Your job 2064447 ("pt_shell") has been submitted
  Stderr   : **<<EMPTY>>**

10] Launched : ...
  Status   : Forking successful
  Stdout   : Your job 2064453 ("pt_shell") has been submitted
  Stderr   : **<<EMPTY>>**
```


Distributed farm creation timeout : 21600 seconds
Waiting for 10 (of 10) distributed hosts (Sat Oct 2 04:12:31 2010)

```

Waiting for 8 (of 10) distributed hosts (Sat Oct 2 04:12:41 2010)
Waiting for 7 (of 10) distributed hosts (Sat Oct 2 04:12:51 2010)
Waiting for 3 (of 10) distributed hosts (Sat Oct 2 04:13:01 2010)
Waiting for 2 (of 10) distributed hosts (Sat Oct 2 04:13:11 2010)
Waiting for 2 (of 10) distributed hosts (Sat Oct 2 04:13:21 2010)
Waiting for 2 (of 10) distributed hosts (Sat Oct 2 04:13:31 2010)
Waiting for 0 (of 10) distributed hosts (Sat Oct 2 04:13:41 2010)
-----
-----

```

1

To report the host options created by the **set_host_options** command as well as the real time data associated with the processes, call the **report_host_usage** command with the **-verbose** option after starting the hosts. Notice that number of the instance launched is reported with the real time data.

```

pt_shell> report_host_usage -verbose
*****
Report : host_usage
      -verbose
Version: D-2009.12
Date   : Tue Nov 24 11:11:39 2009
*****
```

Options Name	Host Name	32Bit	Num Processes
my_opts1	**localhost**	N	1
my_opts2	**localhost**	N	1
my_opts3	pto016	N	1
my_opts4	>>farm<<	N	5
my_opts5	>>farm<<	N	2

Options Name	#	Host Name	Job ID	Process ID	Status
my_opts1	1	pto021	945	945	ONLINE
my_opts2	2	pto021	980	980	ONLINE
my_opts3	3	pto016	20464	20464	ONLINE
my_opts4	4	maddog136	317496	23787	ONLINE
	5	maddog136	317497	23790	ONLINE
	6	maddog136	317498	23789	ONLINE
	7	maddog136	317499	23788	ONLINE
	8	maddog114	317500	21191	ONLINE
my_opts5	9	peopf105	2064447	18524	ONLINE
	10	peopf33	2064453	8798	ONLINE

Options Name	Host Name	Action	Command
my_opts1	**localhost**	SUBMIT	<execute directly on masters host>
my_opts2	**localhost**	SUBMIT	<execute directly on masters host>
my_opts3	pto016	SUBMIT	/usr/bin/rsh -n
my_opts4	>>farm<<	SUBMIT	/lsf/bin/bsub -n 2 -R rusage[mem=500] -
R span[ptile=2]		TERMINATE	/lsf/bin/bkill

```
my_opts5          >>farm<<      SUBMIT      /grid/bin/qsub -b y -P bnormal
                                         TERMINATE /grid/bin/qdel
```

Usage limits (cores)

Options Name	#	User-set	Hardware	Licensed	Effective
(local process)	-	none	4	6	4
my_opts1	1	1	4	1	1
my_opts2	2	1	4	1	1
my_opts3	3	1	4	1	1
my_opts4	4	1	8	1	1
	5	1	8	1	1
	6	1	8	1	1
	7	1	8	1	1
	8	1	8	1	1
my_opts5	9	1	8	1	1
	10	1	8	1	1
Total				12	10

Memory usage

Options Name	#	Memory (MB)
(local process)	-	474.32
my_opts1	1	111.90
my_opts2	2	102.49
my_opts3	3	234.13
my_opts4	4	196.00
	5	108.41
	6	125.83
	7	186.41
	8	178.25
my_opts5	9	233.40
	10	89.83

Performance

Options Name	#	CPU Time (s)	Elapsed Time (s)
(local process)	-	92	180

1

SEE ALSO

```
remove_host_options(2)
set_host_options(2)
start_hosts(2)
stop_hosts(2)
```

report_host_usage

846

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 and pins	Latency				Transition			
	Rise		Fall		Rise		Fall	
	min	max	min	max	min	max	min	max
<hr/>								
-								
middle/in1	1.12	--	1.12	--	--	9.87	--	9.87
middle/p_in	--	--	--	--	--	--	--	--
<hr/>								
Internal pins with ideal timing								
<hr/>								
-								
n3_i/B	--	--	--	--	5.62	5.62	--	--
n0_i/Z	1.34	--	1.34	--	--	--	--	--
<hr/>								
Boundary pins								
<hr/>								
-								
o_reg3/D	--	--	--	--	--	--	--	--
middle/p_out	--	--	--	--	--	--	--	--
<hr/>								
Nets								
<hr/>								
-								
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 and pins	Latency				Transition			
	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_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

<i>library</i>	list
<i>lib_cell_list</i>	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
<hr/>				
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

LTC2

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)

Lib Cell	Attributes	#	Arc	Arc Pins		When
			Sense/Type	From	To	
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
<hr/>				
<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_derrate_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

Cell	Attributes	#	Power		Source of path	When
			Toggling	pin		
<hr/>						
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
OR2	i,a	5	Z			
	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

<i>digits</i>	int
<i>format</i>	string
<i>port_pin_list</i>	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.

```









```

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.

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
```

Pin	Required min period	Actual 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

<i>digits</i>	integer
<i>format</i>	string
<i>port_pin_list</i>	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**.

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)
.in -0.25in
```

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)
```

report_min_pulse_width

```
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 omitted 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
<hr/>			
Uram1/core(RAM2_core)	read(rw)	ENABLED	default
	write(rw)	ENABLED	default
<hr/>			
Uram2/core(RAM2_core)	read(rw)	ENABLED	default
	write(rw)	ENABLED	default
<hr/>			

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

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_multi_input_switching_coefficient

Reports delay and slew coefficients for multi-input switching (MIS) analysis.

SYNTAX

```
int report_multi_input_switching_coefficient  
[-nosplit]  
[-significant_digits num_digits]  
[object_list]
```

Data Types

<i>num_digits</i>	integer
<i>object_list</i>	list

ARGUMENTS

-nosplit	Does not split lines in report when columns overflow.
-significant_digits <i>num_digits</i>	Specifies the number of digits displayed in the report.
object_list	Specifies a list of library cells.

DESCRIPTION

This command reports delay and slew coefficients for multi-input switching (MIS) analysis for the specified list of library cells. If you do not specify a list of library cells, the tool reports the coefficients for all library cells.

EXAMPLES

The following example reports all coefficients set by set_multi_input_switching_coefficient command:

```
pt_shell> report_multi_input_switching_coefficient
```

SEE ALSO

```
set_multi_input_switching_coefficient(2)  
reset_multi_input_switching_coefficient(2)  
si_enable_multi_input_switching_analysis(3)  
si_enable_multi_input_switching_timing_window_filter(3)
```

report_multi_scenario_design

Creates a detailed report on user defined multi-scenario objects and attributes.

SYNTAX

```
Boolean report_multi_scenario_design
  [-scenario]
  [-session]
  [-license]
```

ARGUMENTS

[-scenario]

Reports in detail on the current scenarios that have been defined. Details reported include the name of the scenario, the files used to generate the image for the scenario (i.e. common and specific data files), the scenario used to generate the baseline image for the scenario, the location of the current image, and the focus of the scenario. For detailed description of the settings of the focus field, see the description section below.

[-session]

Reports in detail on the current session. This option reports the number of scenarios defined in the current session. Three fields are reported for each scenario in the current session: the command focus of the scenario, the name of the scenario, and the scenario which provides the image to the scenario.

[-licenses]

Reports in detail on license usage and limits set.

DESCRIPTION

The **report_multi_scenario_design** command generates a report on user-defined multi-scenario analysis. The level of detail reported depends on the options specified. By selecting no options, the **report_multi_scenario_design** command issues a short summary of current user-defined objects and settings. The summary reports on whether or not a session has been defined. The number of defined scenarios and licenses in use are also reported. The report generated can report in detail on the following topics: scenarios, sessions, and licenses.

The *-scenario* option generates a report on all the scenarios that exist in the master process. The report shows several pieces of information about each scenario:

- The name of the scenario
- The files used to generate the images for the scenario (i.e. common and specific data files)
- The scenario used to generate the baseline image for the scenario

- The location of the scenario's current image
- The focus of the scenario

The focus has one of three possible settings:

- "Out of focus" means the scenario is not in the current_session.
- "In current session" means the scenario is in the current session but it is not in command focus. i.e. It will not receive commands from the master.
- "In current session and command focus" means the scenario is in the current session and in command focus. i.e. It will receive commands from the master.

See the **current_scenario** command for changing command focus.

The **-session** option reports the scenarios defined in the current session. Along with the number of scenarios, some scenario-specific data is reported. The focus field of the scenario shows whether the scenario is in the session (Session), or whether it is also in command focus (Command). Also reported is the scenario which provides the image to each scenario.

The **-license** option shows the numbers of licenses checked out for each feature and the user defined limits for each feature.

EXAMPLES

In the following example, several scenarios are created and the current session is set. Then the **report_multi_scenario_design** command is used to examine the multi scenario information available to the master.

```
pt_shell>create_scenario -name scen1 -common_data {com_s1.tcl} -
specific_data {spec_s1.tcl spec_s2.tcl spec_s3.tcl}
1
pt_shell>create_scenario -name scen2 -common_data {com_s2.tcl} -
specific_data {spec_s2.tcl}
1
pt_shell>create_scenario -name scen3 -common_data {com_s3.tcl} -
specific_data {spec_s3.tcl}
1
pt_shell>create_scenario -name scen4 -common_data {com_s3.tcl} -
specific_data {spec_s3.tcl}
1
pt_shell> current_session {scen1 scen2 scen3}
1
pt_shell> report_multi_scenario_design
1
```

```
*****
Report : multi_scenario_design
Version: W-2004.12-Beta3
Date   : Mon Nov  8 08:59:13 2004
*****
```

Summary

```
Current session           : Defined
Number of defined scenarios : 4
Number of scenarios in command focus : 3
```

1

SEE ALSO

```
create_scenario(2)
current_session(2)
current_scenario(2)
set_multi_scenario_license_limit(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

<i>digits</i>	int
<i>net_names</i>	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 $\text{FLT_DIG} - \lceil \log_{10}(\text{fabs}(\text{any_reported_value})) \rceil$.

-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* option 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
<hr/>					
Total nets	30	52	30	56.00	82
Maximum		20	1	25.00	0.00
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 slack_type]
[-all_violators]
[-data_pins]
[-clock_pins]
[-async_pins]
[object_list]
```

Data Types

<i>pin_count</i>	integer
<i>digits</i>	integer
<i>slack_limit</i>	float
<i>object_list</i>	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 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.

-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

<i>digits</i>	integer
<i>from_pin</i>	list
<i>to_pin</i>	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/INVDS3
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
<hr/>				
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
<hr/>				
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
```

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_noiseViolationSources

Reports noise violation sources for failing endpoints.

SYNTAX

```
int report_noiseViolationSources
[-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

<i>pin_count</i>	integer
<i>digits</i>	integer
<i>slack_type</i>	float
<i>object_list</i>	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 leveled 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_noiseViolationSources -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_noiseViolationSources -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

```
string report_ocvm
[-early]
[-late]
[-rise]
[-fall]
[-clock]
[-data]
[-voltage voltage_value]
[-cell_delay]
[-net_delay]
[-list_annotated]
[-list_not_annotated]
[-coefficient]
-type aocvm | pocvm
[-nosplit]
[object_list]
```

Data Types

<i>voltage_value</i>	string
<i>object_list</i>	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.

-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/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_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, maxLoad, 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

Port	Dir	Pin	Wire	Attributes
		Cap	Cap	
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]
[object_list]
```

Data Types

<i>low_vth_list</i>	list
<i>number</i>	integer
<i>sort_method</i>	string
<i>level</i>	integer
<i>threshold</i>	float
<i>clock_list</i>	list
<i>group_list</i>	list
<i>rails_supply_nets_name</i>	list
<i>pattern_list</i>	list
<i>attribute_name_for_cell_pattern</i>	string
<i>object_list</i>	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,
- report_power
900

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.

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
```

Net	Vdd	Total Net Load	Static Prob.	Toggle Rate	Switching 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   %
-----
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
```

Power Group	Internal Power	Switching Power	Leakage Power	Total 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
```

```
report_power
```

```
912
```

Attributes

i - Including register clock pin internal power
u - User-defined power group

Power Group	Internal Power	Switching Power	Leakage Power	Total 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

Power Group	Internal Power	Switching Power	Leakage Power	Total 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\(2\)](#)

[report_power](#)

914

```
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

This command has no arguments.

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

```
int report_power_calculation
[-state_condition state]
[-path_sources name_list]
[-rise]
[-fall]
[-verbose]
object_list
```

Data Types

state	string
name_list	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.

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.

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** command included in the library, which enables the **report_power_calculation** command for internal, leakage, and net switching power. Otherwise, the built-in security mechanism terminates the command.

The **report_power_calculation** command is only enabled in averaged power analysis mode. For more information about analysis mode, see the **power_analysis_mode** man page.

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

report_power_calculation
920

```
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

<i>object_list</i>	list
<i>digits</i>	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
*****
```

Subth	Object	Name	Object Type	Switching	Internal	Leakage	Leaka
reshold	Gate						
ge	Object	Name	Type	Switching	Internal	Leakage	Leaka
ge	Leakage						
-----	-----	-----	-----	-----	-----	-----	-----
-----	top	design		0.92	1.15	--	
1			1.26	0.75			

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
*****
```

shold	Gate				Subthre	
Object	Name	Object Type	Switching	Internal	Leakage	Leakage
		Leakage				
<hr/>						
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 for UPF

```
int 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 : PDO
Scope : <top level>
Elements : PDO_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_domain
928

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_net_info

Reports the power net info for the current design.

SYNTAX

```
int report_power_net_info
```

ARGUMENT

The **report_power_net_info** command has no arguments.

DESCRIPTION

The **report_power_net_info** command reports the power net info for the current design.

EXAMPLE

The following example uses the command to report the power net info.

```
pt_shell> create_power_domain T  
1  
pt_shell> create_power_net_info T_VDD -power  
1  
pt_shell> create_power_net_info T_VSS -gnd  
1  
pt_shell> create_power_domain T  
1  
pt_shell> connect_power_domain T \  
      -primary_power_net T_VDD \  
      -primary_ground_net T_VSS  
1  
prompt> report_power_net_info
```

```
*****  
Report : power_net  
Design : top  
Version: Z-2007.06-DEV  
Date   : Wed May 16 15:18:42 2007  
*****
```

Total of 2 power nets defined.

```
Power Net 'T_VDD' (power)  
-----  
Primary Power Hookups:          T  
Internal Power Hookups:         T  
-----
```

```
Power Net 'T_VSS' (ground)  
-----  
Primary Ground Hookups:          T
```

Internal Ground Hookups:

T

1

SEE ALSO

`create_power_net_info(2)`
`connect_power_domain(2)`

report_power_network

Reports connectivity in the virtual power network formed by UPF supply nets, ports, and PG pins.

SYNTAX

```
string report_power_network
[-nets nets]
```

Data Types

nets string

ARGUMENTS

```
-nets nets
      Reports explicit connectivity of a subset of nets.
```

DESCRIPTION

This command reports the explicit connectivity of supply nets, ports, and connected power and ground pins.

EXAMPLES

The following example shows the explicit connections of the given supply net, VDDI when you specify the **connect_supply_net** command.

```
prompt> report_power_network -nets VDDI
```

```
Supply Net: VDDI
Explicit Connections:
  Name :          Object Type        Domain
-----
  VDDI           Supply Port       TOP
  InstDecode/LS_u1/VDDL PG_power_pin   INST
in -0.25in
```

SEE ALSO

create_supply_net(2)
report_supply_net(2)

report_power_pin_info

Reports the power pin information for technology library cells or leaf cells.

SYNTAX

```
int report_power_pin_info  
object_list
```

Data Types

object_list list

ARGUMENT

object_list
Specifies a list of technology library cells or a list of leaf cells or ports.

DESCRIPTION

The **report_power_pin_info** command reports the power pin information for technology library cells or leaf level cells in the current design.

When a list of library cells is specified, 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.

When a list of leaf cells are specified in the *object_list* variable, 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 -hier]
```

```
*****  
Report : power pin info  
Design : top  
Version: Z-2007.06-DEV  
Date   : Wed May 16 15:18:42 2007  
*****
```

Note: Power connections marked by (*) are exceptional

Cell	Power Pin Name	Type	Voltage	Powe
r Net Connected				
PDO_INST/IO	PWR	primary_power	1.0000	

PDO_INST/I0	GND	primary_ground	0.0000
PDO_INST/I1	PWR	primary_power	1.0000
PDO_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 [get_cells -hier]]
```

```
*****
```

Report : power pin info

Design : top

Version: Z-2007.06-DEV

Date : Wed May 16 15:18:42 2007

```
*****
```

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 PDO_INST/I0 -power_pin_name PWR -
power_net_name exp_VDD
```

1

```
pt_shell> report_power_pin_info [get_cells -hier]
```

```
*****
```

Report : power pin info

Design : top

Version: Z-2007.06-DEV

Date : Wed May 16 15:18:42 2007

```
*****
```

Note: Power connections marked by (*) are exceptional

Cell r Net Connected	Power Pin Name	Type	Voltage	Powe
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 -hier]
```

```
*****
Report : power pin info
Design : top
Version: Z-2007.06-DEV
Date   : Wed May 16 15:18:42 2007
*****
```

Note: Power connections marked by (*) are exceptional

Cell r Net Connected	Power Pin Name	Type	Voltage	Powe
PDO_INST/				
I0	PWR	primary_power	1.0000	exp_VDD (*)
PDO_INST/I0	GND	primary_ground	0.0000	A_VSS
PDO_INST/I1	PWR	primary_power	1.0000	A_VDD
PDO_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

Report the current power rail mapping.

SYNTAX

```
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 its column's 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 doesn't 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

The following example shows 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
```

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_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

<i>digits</i>	int
<i>port_pin_list</i>	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	
<hr/>				
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`

Pin	Required Transition	Actual Transition	Slack	
<hr/>				
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_transition`

942

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

<i>digits</i>	int
<i>format</i>	string
<i>port_pin_list</i>	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

<i>digits</i>	int
<i>port_pin_list</i>	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

Pin	Required Transition	Actual Transition	Slack
p1/i	0.30	0.12	-0.18 (VIOLATED)

pulse_clock_min_transition_fall

Pin	Required Transition	Actual 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

<i>digits</i>	int
<i>format</i>	string
<i>port_pin_list</i>	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
```

Pin	Required pulse width	Actual 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
```

Pin	Required pulse width	Actual 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

```
string report_qor
[-max]
[-min]
[-significant_digits digits]
[-only_violated]
[-summary]
[-physical]
```

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.

-physical

This option has no effect in PrimeTime.

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**.

Limitations

The command does not provide a merged distributed multi-scenario analysis (DMSA) report. Executing the command on the DMSA master produces an error message. If you use DMSA, use the **report_qor** Tcl procedure available in SolvNet article 008602.

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 -only_violated

*****
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)
```

report_qor
952

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
------	------	-----------	------------	--------	-----------------

report_qtm_model

954

```
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_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

Use to report the scaling that was done previously using the `scale_parasitics` command.

SYNTAX

```
int report_scale_parasitics
[net_list]
```

Data Types

net_list list

ARGUMENTS

net_list

Limits the report of scaling to this list of nets. If this option is used, it is assumed that net-specific scaling has been done for those nets. If this is not the case, an error gets generated.

DESCRIPTION

The `report_scale_parasitics` command can be used to report the scale factors for parasitic-scaling that was done previously through the `scale_parasitics` command.

EXAMPLES

This example reports scaling of parasitics done previously.

```
pt_shell> report_scale_parasitics
```

SEE ALSO

```
read_parasitics(2)
scale_parasitics(2)
reset_scale_parasitics(2)
```

report_scope_data

Reports relevant scope data stored in the specified file.

SYNTAX

```
int report_scope_data
[-block_name blk_name]
[-port_names block_level_port_names]
[-clock_names block_level_clock_names]
[-scope_scenarios scenario_names]
[-check_types check_types]
[-instance instance_name]
[-significant_digits digits]
[-nosplit]
[-verbose]
file_name
```

Data Types

<i>blk_name</i>	string
<i>block_level_port_names</i>	string
<i>block_level_clock_names</i>	string
<i>scenario_names</i>	string
<i>check_types</i>	string
<i>instance_name</i>	string
<i>digits</i>	int

ARGUMENTS

-block_name *blk_name*

Specifies the name of the block for which the scope data is to be reported. By default, if not specified, scope data for all blocks stored in the file is reported.

-port_names *block_level_port_names*

Specifies the names of ports for which the scope data is to be reported. Note these names are the block-level port names. If not specified, all input port scope data in the file is reported.

-clock_names *block_level_clock_names*

Specifies the names of clocks for which the scope data is to be reported. Note these names are for the block-level clocks. If not specified, all clock scope data in the file is reported.

-scope_scenarios *scenario_names*

Specifies the list of scenario names for which the scope data are to be reported. If not specified, all available scenarios in the scope data file are reported.

-check_types *check_types*

Specifies the types of scope data to be reported from the file. If not specified, the types defined by environment variable

hier_scope_check_defaults are used. Note that check type *clock_skew_with_uncertainty* is a derived type so it is not supported by this command.

-instance *instance_name*
 Specifies the name of a cell instance in the design for which the scope data is to be reported. This requires that the top level design to be properly linked.

-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.

-nosplit
 Specifies whether the long lines in the report should be split into multiple lines. By default, lines are split.

-verbose
 Indicates that the report of scope data should be very detailed. If not specified, by default, only a very summarized report is given for the scope data stored in the file. However, when one of the options **-clock_names**, **-port_names**, **-check_types** is specified, the report will always be verbose with or without **-verbose**.

file_name
 Specifies the file that contains the scope data.

DESCRIPTION

This command is used to report on the information stored in the binary scope data file. It provides users with convenient access to the detailed but encrypted scope information. It is for data reporting purpose only, no modification is done to the existing data, nor any computation or checking is done.

Because it is just report the data from the file, this command can be used with or without any designs loaded and linked in memory. However, if option **-instance** is used, a linked top-level design is required.

EXAMPLES

This example generates a summarized reports of the available scope data stored in file *top/ilm.scope*.

```
pt_shell> report_scope_data top/ilm.scope
```

The following command generates a very detailed reports of the scope data stored in file *my_ETM_block.scope*

```
pt_shell> report_scope_data -verbose my_ETM_block.scope
```

SEE ALSO

```
create_ilm(2)
extract_model(2)
create_scenario(2)
check_block_scope(2)
update_scope_data(2)
hier_scope_check_defaults(3)
```

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

<i>clock_list</i>	list
<i>object_list</i>	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.

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
```

```

...
*****
Object          Clock          Sense
-----
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

```
int report_si_aggressor_exclusion
[-rise]
[-fall]
[-nosplit]
[anets]
```

Data Types

anets 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.

anets

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.

-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_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.

report_si_aggressor_exclusion

964

```

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
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
Agg1 Agg2 Agg3	R	2

```

pt_shell> report_si_aggressor_exclusion [get_nets Agg2] -fall
*****
Report : report_si_aggressor_exclusion
         -rise
         -fall
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
Agg2	Agg4	1

SEE ALSO

set_si_aggressor_exclusion(2)
remove_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)

report_si_bottleneck

Identify the crosstalk bottlenecks in the design. This is useful when the major sources of violations come from crosstalk effects.

SYNTAX

```
int report_si_bottleneck
[-cost_type type]
[-slack_lesser_than slack]
[-max_nets number]
[-significant_digits digits]
[-nosplit]
[-include_clock_nets]
[-minimum_active_aggressors number]
[-min]
[-max]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
```

Data Types

<i>type</i>	string
<i>slack</i>	float
<i>count</i>	int
<i>digits</i>	int
<i>active_aggressor_count</i>	int
<i>pre_command_string</i> "	string
<i>post_command_string</i> "	string

ARGUMENTS

```
[-cost_type type]
    The nets are sorted based on the cost. The values available for the type option are delta_delay, delta_delay_ratio, and total_victim_delay_bump to identify the cost of the victim net and delay_bump_per_aggressor for aggressor nets of the selected victim nets.

[-slack_lesser_than slack]
    The cost function is applied only to the victim nets whose slack is less than the slack limit.

[-max_nets number]
    The maximum number of nets to be reported. The default is 20.

[-significant_digits digits]
    The number of digits to display: Range: 0 to 13. The default is the same as the report_default_significant_digits global variable.

[-nosplit]
    Don't split lines if column overflows.
```

```

[-include_clock_nets]
    Include the clock nets for bottleneck. By default clock nets are excluded
    from the bottleneck analysis.

[-minimum_active_aggressors number]
    The minimum number of active aggressors for the net to be selected. The
    default is 1.

[-min]
    For the minimum analysis. For example, the slack limit applies to minimum
    slack of the net in the net selection.

[-max]
    For the maximum analysis. For example, the slack limit applies to maximum
    slack of the net in the net selection.

[-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 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.

Four cost functions are available. 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 aggresses, 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.

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 may 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, the user could potentially 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, thus preventing the cost of an additional timing update.

EXAMPLES

The following examples show a PrimeTime SI bottleneck report. arc with specified start and end points.

```
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 -aggressors [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 Net	Switching Direction	Actual Bump Height	Required Bump Height	Double Switching 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 -aggressors [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
*	LOGICO	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_supply_net

Reports on existing supply nets.

SYNTAX

```
int report_supply_net
[-domain domain_name]
[-scope scope_name]
```

Data Types

<i>domain_name</i>	string
<i>scope_name</i>	string

ARGUMENTS

-domain *domain_name*
Specify domain dependent supply nets which exist in a particular power domain. The domain must exist.

-scope *scope_name*
Specify for which scope you want to define the supply nets that are reported. If this is not specified, report_supply_net reports all the power nets in the current scope.

DESCRIPTION

The **report_supply_net** command enables you to get the detail information of all supply nets associated with specified power domain or scope. If a domain is specified, only the domain dependent supply nets that exist in the domain are reported. The information of supply nets include: its full name, all power domain it is created in, all voltage information that is explicitly set on this net by the **set_voltage** command. Also, listed are all domains for which the net is the primary power or ground net.

The command returns 1 if it is successful; otherwise, 0 is returned.

EXAMPLES

The following example reports the information of all supply nets in the current scope:

```
prompt> report_supply_net
      Total of 1 supply nets defined.
-----
Supply Net :      i2/net_i2
Scope :          i2
Power Domains :
Supply Ports   :      N/A
PG_power_pin  :      N/A
```

```
Max-delay Voltage : 1.08
Min-delay Voltage : 1.3
Resolve type : unresolved
1
```

SEE ALSO

```
create_supply_net(2)
create_power_domain(2)
connect_supply_net(2)
get_power_domains(2)
set_domain_supply_net(2)
set_scope(2)
```

report_supply_set

Reports all supply sets defined in the design.

SYNTAX

```
status report_supply_set
```

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.

ARGUMENTS

This command has no arguments.

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_switching_activity

Reports the statistics on the switching activity annotation and signal probability on the current design or instance.

SYNTAX

```
int report_switching_activity
[-cells cell_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 | toggle]]
[-toggle_limit limit]
[-exclude exclusion_group]
[-include_only inclusion_group]
[-only_related_clock clock]
```

Data Types

<i>cell_list</i>	list
<i>source</i>	string
<i>clk</i>	string
<i>limit</i>	int
<i>exclusion_group</i>	string
<i>inclusion_group</i>	string
<i>clock</i>	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 (vcf 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 *impliedP* 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"
-----
```

Object (%)	Type	From Activity	From	From	From	Default
		File (%)	Propagated (%)	Implied (%)	Not SSA (%)	
				Annotated (%)	Total	

Nets	1613 (99.02%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	0 (0.00%)	0 (0.00%)	16 (0.98%)	1629	

Nets Driven by

Primary Input	67 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	0 (0.00%)	0 (0.00%)	0 (0.00%)	67	
Tri-State	0 (0%)	0 (0%)	0 (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%)	0 (0%)	0 (0%)	0		
Sequential	209 (92.89%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
0 (0.00%)	0 (0.00%)	16 (7.11%)	225		
Combinational	1337 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
0 (0.00%)	0 (0.00%)	0 (0.00%)	1337		
Memory	0 (0%)	0 (0%)	0 (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%)	0 (0%)	0 (0%)	0		

Static Probability Overview Statistics for "mac"

Object (%)	Type	From Activity		From Not SSA (%)	From SCA (%)	From Clock (%)	From Default Total
		File (%)	Propagated (%)	Implied (%)			
Nets		1613 (99.02%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
		0 (0.00%)	0 (0.00%)	16 (0.98%)	1629		

Nets Driven by

Primary Input	67 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
	0 (0.00%)	0 (0.00%)	0 (0.00%)	67	
Tri-State	0 (0%)	0 (0%)	0 (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%)	0 (0%)	0 (0%)	0		
Sequential	209 (92.89%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
0 (0.00%)	0 (0.00%)	16 (7.11%)	225		
Combinational	1337 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
0 (0.00%)	0 (0.00%)	0 (0.00%)	1337		
Memory	0 (0%)	0 (0%)	0 (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%)	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"

Object (%)	Type	From Activity	From	From	From	
		File (%) Propagated(%)	Not Implied(%)	SSA (%) Annotated(%)	SCA (%) Total	Clock (%) Default
Nets		1613 (99.02%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 16 (0.98%)	0 (0.00%) 1629	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%)	0 (0.00%) 67	0 (0.00%) 0 (0.00%)
Tri-State	0 (0%) 0 (0%)	0 (0%) 0 (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%)	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%)	0 (0.00%) 225	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%)	0 (0.00%) 1337	0 (0.00%) 0 (0.00%)
Memory	0 (0%) 0 (0%)	0 (0%) 0 (0%)	0 (0%) 0 (0%)	0 (0%) 0	0 (0%) 0 (0%)
Clock Gate	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)

report_switching_activity
994

0 (0%) 0 (0%) 0 (0%) 0

Static Probability Overview Statistics for "mac"

Object (%)	Type	From Activity		From Not SSA (%)	From SCA (%)	From Clock (%)	From Default Total
		File (%)	Propagated(%) Implied(%)				
Nets		1613 (99.02%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
		0 (0.00%)	0 (0.00%)	16 (0.98%)	1629		
<hr/>							
Nets Driven by							
Primary	Input	67 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
		0 (0.00%)	0 (0.00%)	0 (0.00%)	67		
<hr/>							
Tri-State		0 (0%)	0 (0%)	0 (0%)	0 (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%)
		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%)
		0 (0.00%)	0 (0.00%)	16 (7.11%)	225		
Combinational		1337 (100.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
		0 (0.00%)	0 (0.00%)	0 (0.00%)	1337		
Memory		0 (0%)	0 (0%)	0 (0%)	0 (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%)
		0 (0%)	0 (0%)	0 (0%)	0		
<hr/>							

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"
```

Object (%)	Type	From Activity		From Not SSA (%)	From SCA (%)	From Clock (%)	From Default Total
		File (%)	Propagated(%) Implied(%)				
Nets		1613 (99.02%)		0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
		16 (0.98%)	0 (0.00%)	0 (0.00%)		1629	

```
Nets Driven by
```

Primary State	Input 0 (0%)	67 (100.00%)	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	0 (0%)	209 (92.89%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Combinational	0 (0.00%)	1337 (100.00%)	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"
```

```
*****
```

```
report_switching_activity
```

```
996
```

Object (%)	Type	File (%)	SSA (%)	SCA (%)	Clock (%)	Default
		Propagated (%)	Implied (%)	Annotated (%)	Total	
<hr/>						
Nets		1613 (99.02%) 16 (0.98%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 1629	0 (0.00%)
<hr/>						
Nets Driven by						
<hr/>						
Primary Input		67 (100.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 67	0 (0.00%)
<hr/>						
Tri-State		0 (0%) 0 (0%)	0 (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%)	0 (0%) 0	0 (0%)
Sequential		209 (92.89%) 16 (7.11%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 225	0 (0.00%)
Combinational		1337 (100.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 0 (0.00%)	0 (0.00%) 1337	0 (0.00%)
Memory		0 (0%) 0 (0%)	0 (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%)	0 (0%) 0	0 (0%)
<hr/>						

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 : counter
Version: V-2004.06-Alpha1
Date   : Mon Mar 29 11:09:34 2004
*****
```

```
Switching Activity Average for "counter"
```

```
Switching Activities per clock is with respect to clock 'CLK'
Simulation time 10us (1000 clock periods of clock 'CLK')
```

Object	Toggle Count	Toggle Rate	Glitch Count	Glitch Rate
--------	--------------	-------------	--------------	-------------

	Per Net	Per Clock	Per Net	Per Clock
		Per Net		Per Net
counter	200	0.2	1	0.001
counter_low_bits	150	0.15	1	0.001
counter_high_bits	50	0.05	0	0
reset_block	1	0.001	0	0
overflow_detect	1	0.001	0	0

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_switching_activity

998

```
report_power(2)
```

```
report_switching_activity  
999
```

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]
[-pattern_priority pattern_list]
[-attribute attribute_name_for_cell_pattern]
[-nosplit]
[-verbose]
[-area]
cell_list_or_current_design
```

Data Types

<i>groups</i>	list
<i>cell_list_or_current_design</i>	list
<i>pattern_list</i>	list
<i>attribute_name_for_cell_pattern</i>	string

ARGUMENTS

-lvth_groups *groups*

Specifies the list of threshold voltage groups to be 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.

-pattern_priority *pattern_lists*

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.

-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.

report_threshold_voltage_group

1000

-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 ag190g_od_svt_ss] default_threshold_voltage_
group SVT
pt_shell> set_user_attribute [get_libs ag190g_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 Name	ABCD cell (%)	LVT(L) cell (%)	XYZK(L) cell (%)	SVT 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_threshold_voltage_group
1002

report_timing

Reports timing paths.

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]
[-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]
[-report_ignored_register_feedback]
[-significant_digits digits]
[-crosstalk_delta]
[-derate]
[-variation]
[-exceptions exception_type]
[-voltage]
[-sort_by group | slack]
[path_collection]
```

Data Types

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>exclude_list</i>	list
<i>rise_exclude_list</i>	list
<i>fall_exclude_list</i>	list
<i>delay_type</i>	string
<i>paths_per_endpoint</i>	integer
<i>max_path_count</i>	integer
<i>group_name</i>	list
<i>minimum_slack</i>	float
<i>maximum_slack</i>	float
<i>feedback_slack_cutoff</i>	float
<i>pre_command_string</i>	string
<i>post_command_string</i>	string
<i>format</i>	string
<i>digits</i>	integer
<i>exception_type</i>	string
<i>path_collection</i>	collection

ARGUMENTS

-from *from_list*

Reports only paths from the named pins, ports, nets, cell instances, or startpoints clocked by named clocks. If *from_list* is not specified, the default behavior reports the longest path to an output port if the design has no timing constraints. Otherwise, the default behavior is to report the path with the worst slack if the design has timing constraints.

-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*

Reports only paths to the named pins, ports, nets, cell instances, or endpoints clocked by named clocks. If *to_list* is not specified, the default behavior reports the longest path to an output port if the design has no

timing constraints. Otherwise, the default behavior is to report the path with the worst slack if the design has timing constraints. If a clock object is to be specified with the **-to** option, it must be specified as a collection (that is, using **[get_clocks clock_name]**).

-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*

Reports only data paths through the named pins, ports, cell instances, or nets. You cannot use this option to report clock paths. If the *through_list* option is not specified, the default behavior reports the longest path to an output port if the design has no timing constraints. Otherwise, the default behavior reports the path with the worst slack if the design has timing constraints.

If advanced analysis through transparent latches is enabled

(**timing_enable_through_paths** is 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 edge at the latch or by a required value computed using logic downstream of the latch loop breaker. The latter constraint is referred to as 'time required through endpoint' in the timing report and the latch is said to be constrained by downstream required.

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 *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 *through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects.

-exclude *exclude_list*

Prevents the reporting of data paths that contain the specified pins, ports, nets, and cell instances. Reporting excludes all data paths from, through,

or to the named pins, ports, nets, and cell instances. If a cell instance is specified, all pins of the cell are excluded. The **-exclude** option

- Takes precedence over the **-from**, **-through**, and **-to** options.
- Is exclusive with the **-rise_exclude** and **-fall_exclude** options.
- Does not apply to borrowing path from the **-trace_latch_borrow** option or clock path from the **-path full_clock** or **full_clock_expanded** option.
- Does not apply to clock pins.

-rise_exclude *rise_exclude_list*

Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, and cell instances.

-fall_exclude *fall_exclude_list*

Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, and cell instances.

-delay_type *delay_type*

Specifies one of the following path delay to be reported, where "rise" or "fall" indicates a rising or falling transition at the path endpoint:

- **max** (the default)
- **min**
- **min_max**
- **max_rise**
- **max_fall**
- **min_rise**
- **min_fall**

-nworst *paths_per_endpoint*

Reports the specified number of worst paths per endpoint. Allowed values are 1 to 2000000; the default is 1.

-max_paths *max_path_count*

Reports up to the specified maximum total number of paths among all path groups, where $1 \leq max_path_count \leq 2000000$; the default is equal to the **-nworst** setting. If you specify *max_path_count* > 1, and you do not explicitly specify either the **-slack_lesser_than** or the **-slack_greater_than** option, the tool automatically sets **-slack_lesser_than 0** and reports only violating paths.

-group *group_name*

Specifies the path groups from which timing paths are selected for reporting based on other specified options for reports. Use this option to report the worst paths in each path group, for example, by using **report_timing -group [get_path_groups *]**.

-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. For non-unate logic such as XOR gates, this greatly reduces the number of paths reported because of the large number of possible rising/falling edge combinations through the 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 (more positive) than the *minimum_slack* value. This option is applied as a filter to the paths after they are generated. Therefore, the number of paths generated might be fewer than the number specified with the **-nworst** and **-max_paths** options. To show paths within a specific slack range, use this option together with the **-slack_lesser_than** option.

-slack_lesser_than *maximum_slack*

Reports only paths with slack less (more negative) than the *maximum_slack* value. To show paths within a specific slack range, use this option together with the **-slack_greater_than** option. If you specify *max_path_count* > 1, and you do not explicitly specify either the **-slack_lesser_than** or the **-slack_greater_than** option, the tool automatically sets **-slack_lesser_than 0** and reports only violating paths.

-ignore_register_feedback *feedback_slack_cutoff*

Ignores noninverting timing loops if they 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 either both be inverting or both be noninverting. This option applies to minimum delay as well as maximum delay reports. Paths are ignored only if they have a slack less than the specified *feedback_slack_cutoff* value. 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.

-include_hierarchical_pins

Includes hierarchical pins in the report.

-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 entire set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. By default, the report shows only the last path in the sequence of borrowing path segments. Each path segment is reported separately, showing the time borrowed and lent and the endpoints of the path segment. The cumulative amount of borrowed time along a sequence of path segments is not included in the report. The **-input_pins**, **-nets**, **-transition_times**, **-capacitance**, and **-significant_digits** options apply to every path segment in the sequence of borrowing paths, but the remaining options (for example, the **-from** option) apply only to the last path segment

reported. If you use this option with the **-trace_latch_forward** option, the trace forward path segments are appended to the last path segment.

-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 downstream required then 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 downstream required. In this case, each path segment is reported separately showing the downstream required for that path segment. The **-input_pins**, **-nets**, **-transition_times**, **-capacitance**, and **-significant_digits** options apply to every path segment in the timing report. The remaining options apply only to the first path segment reported. If you use this option with the **-trace_latch_borrow** option, the trace borrow path segments are prepended to the first path segment.

-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 to paths after they have been gathered.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis path set in the design. If you use **exhaustive**, you cannot use the **-start_end_pair**, **-cover_design**, or **path_collection** options.

-normalized_slack

Paths are gathered and sorted using normalized slack instead of slack. Normalized slack divides the slack by an idealized allowed propagation delay. In order to use this option, you must run **update_timing** with the **timing_enable_normalized_slack** variable set to true.

-start_end_pair

Reports paths for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtime and can lead to generating a huge number of paths depending on the design. By default, this option searches only for paths that are violating. This default can be changed by having an explicit **-slack_lesser_than** option. The options that do not work with this option are **-nworst**, **-max_paths**, **-unique_pins**, **-slack_greater_than**, **-ignore_register_feedback**, **-report_ignored_register_feedback**, **-cover_design**. Unlike other options of the **report_timing** command, this option causes the paths reported to no longer be sorted based on slack. Instead, paths are arranged based on the endpoint with those sharing the same endpoint appearing next to one another. The maximum number of paths reported is limited to 2000000. To avoid the potential of returning duplicate paths, this option works as though the **timing_report_always_use_valid_start_end_points** variable was set to true.

`-cover_design`

Reports the worst path through each violating pin in the design. A pin is deemed to be violating if its slack is less than the value specified with **-slack_lesser_than**. The options that do not work with this option are **-nworst**, **-max_paths**, **-unique_pins**, **-ignore_register_feedback**, **-report_ignored_register_feedback**, **-start_end_pair**, **-pba_mode exhaustive**. If you use the **-pba_mode path** option, the paths are gathered according to the graph-based-analysis slack and recalculated. To control the path ordering and slack filtering (GBA or PBA slack), use the **pba_path_mode_sort_by_gba_slack** variable. The maximum number of paths reported is limited to 2000000.

`-dont_merge_duplicates`

This option is available only if you invoke PrimeTime with the **-multi_scenario** option. It turns OFF a main capability in merged reporting that is ON by default. The option affects the manner in which paths from multiple scenarios are merged. By default, when the same path is reported from more than one scenario, PrimeTime reports only the single most critical instance of that path in the merged report and shows its associated scenario. When you use this option, PrimeTime does not merge duplicate instances of the same path into a single instance, but instead shows all critical instances of the path from all scenarios. Since the number of paths reported is limited by the **-nworst**, **-max_paths**, and other options of this command, the resulting merged report, when this option is used, might not be evenly spread out across the design, but instead can be focused on the portion of the design that is critical in each scenario.

`-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 commands. Commands must be grouped using the semicolon (;) 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.

`-path_type format`

Specifies the format of the path report and how the timing path is displayed. The allowed values are

- **short** - Displays only start and endpoints in the timing path.
- **full** (the default) - Displays the full path.
- **full_clock** - Displays full clock paths in addition to the full timing path.
- **full_clock_expanded** - Displays full clock paths between a primary clock and a related generated clock in addition to the full_clock timing path.
- **end** - Generates a report with a column format that shows one line for each path and shows only the endpoint path total, required-time, slack, and CRP

(clock reconvergence pessimism value) when the **timing_remove_clock_reconvergence_pessimism** variable is set to **true**.

- **summary** - Shows a summary of the total slack between the startpoint and endpoint.

-input_pins
Shows input pins in the path report. By default, the report shows only output pins.

-nets
Shows nets in the path report. By default, the report does not show nets.

-nosplit
Prevents line splitting, which is useful for scripts that extract information from the report output.

-transition_time
Shows the transition time (slew) in the path report. 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 capacitance in the path report. The default is not to show capacitance. The capacitance value printed depends on the **report_capacitance_use_ccs_receiver_model** variable setting. If **report_capacitance_use_ccs_receiver_model** is set to **true** (the default), then the CCS receiver capacitance is reported. If the **report_capacitance_use_ccs_receiver_model** variable is set to **false**, then the lumped capacitance is reported. For each driver pin the total capacitance driven by the driver is displayed in a column preceding both incremental path delay and transition time (with **-transition_time**). When the **-nets** option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

-report_ignored_register_feedback
Reports ignored paths when you specify the **-ignore_register_feedback** option.

-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 defaults to 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.

-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.

-derate
Shows derating factors in the timing report. The default is to not show derate factors. Specifying this option automatically sets the **-input_pins** option, so that different net and cell derates can be distinguished. When **full_clock_expanded** path types are reported an additional summary report follows the timing report showing the overall effect of derating the path. Derate factors are always shown with at least two significant digits.

-variation
Includes variation information in the report.

-exceptions dominant | overridden | all
Prints user-specified timing exceptions, namely false paths, multicycle paths, and minimum and maximum delays, that are satisfied per timing path being reported. The reason of an unconstrained timing path is also printed.
Note: The additional analysis required per path with the **-exceptions** option is not trivial. Therefore, using a **report_timing** command with the **-exceptions** option is expected to execute slower than the exact same command without this option. The **-exceptions** option does not work with **-path_type short/end/summary** option and forces the **-input_pins** option.

-voltage
Reports the operating voltage for each path element. This option allows to debug voltages in multivoltage designs. The displayed voltage is the voltage used in delay calculation - typically the voltage in operating condition, voltage of the related supply net or power pin, or pin signal level.

-sort_by group | slack
Specifies the sorting of the reported paths.

- **slack** (default) - Sorts paths by slack.
- **group** - Sorts paths by path group. Within each group, the paths are ordered by slack. Using the **group** option is the equivalent of using **report_timing -group [get_path_groups *] -sort_by slack**.

path_collection
Specifies the collection of timing paths to report. When you use this option with the **-pba_mode path** option, path-based analysis is performed on the paths in the *path_collection* before they are printed.
This option is mutually exclusive with options that control the selection of paths to report and is only compatible with options which control the formatting of the report.

DESCRIPTION

The **report_timing** command reports timing information for the current design.

Back-annotations on path elements in the timing path are indicated by a character symbol. If the **-input_pins** option is used, each pin-to-pin delay spans either a net or cell. The symbol shown indicates the dominant annotation on this path element. (Certain annotations dominate others; for example, SDF takes precedence over back-

annotated RC parasitics.) Without the **-input_pins** option, the delay shown could span both a net and a cell. If they 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.

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

The symbol indicates only the dominant annotation; you should note that this might not have been used to calculate the observed delay. For example, suppose that a back-annotated RC network exists on a net, but that the network calculation fails to converge for the driver cell delay. In this case, you are alerted, and a lumped RC model is used instead; but, during report timing, the "&" symbol appears anyway. To see how the pin-to-pin delay is actually calculated, use the **report_delay_calculation** command.

The command can also report on a collection of timing paths which were retrieved using the **get_timing_paths** command.

When the **-exceptions** option is used, see the following possible reasons for unconstrained paths and their corresponding timing path attributes (the timing path attributes are not contained in the report):

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

The following example shows a timing report using only the default settings:

```
pt_shell> report_timing
*****
Report : timing
    -path_type full
    -delay max
    -max_paths 1
...
*****
```

```

Startpoint: c (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

```

Point	Incr	Path
<hr/>		
input external delay	0.00	0.00 r
c (in)	0.00	0.00 r
u1/Z (IVA)	0.54	0.54 f
u0/Z (NR2)	1.20	1.74 r
u8/Z (IVA)	0.43	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
<hr/>		
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00
<hr/>		
data required time		0.00
data arrival time		-3.41
<hr/>		
slack (VIOLATED)		-3.41

The following example reports the endpoint path delay, required time, and slack for each path:

```
pt_shell> report_timing -path_type end
```

```
*****
Report : timing
  -path_type end
  -delay max
...
*****
```

Endpoint	Path Delay	Path Required	Slack
<hr/>			
z2	3.41 f	0.00	-3.41
z3	3.03 f	0.00	-3.03
z4	2.77 f	0.00	-2.77
z6	2.69 r	0.00	-2.69
z0	2.59 f	0.00	-2.59
z1	2.37 f	0.00	-2.37
z5	2.26 f	0.00	-2.26

The following example reports the start and endpoints of the path from a to z2:

```
pt_shell> report_timing -from a -to z2 -path_type short
```

```
*****
Report : timing
  -path_type short
  -delay max
```

```

-max_paths 1
...
*****
Startpoint: a (input port)
Endpoint: z2 (output port)
Path Group: default
Path Type: max

Point           Incr      Path
-----
input external delay    0.00    0.00 f
a (in)            0.00    0.00 f
...
z2 (out)          1.24    1.24 f
data arrival time           1.24

max_delay        0.00    0.00
output external delay   0.00    0.00
data required time           0.00
-----
data required time        0.00
data arrival time         -1.24
-----
slack (VIOLATED)        -1.24

```

The following example shows input pins in the report, in addition to the default values:

```
pt_shell> report_timing -input_pins
```

```

*****
Report : timing
  -path_type full
  -delay max
  -input_pins
  -max_paths 1
...
*****

Startpoint: c (input port)
Endpoint: z2 (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
u1/A (IVA)        0.00    0.00 r
u1/Z (IVA)        0.54    0.54 f
u0/A (NR2)         0.00    0.54 f
u0/Z (NR2)         1.20    1.74 r
u8/A (IVA)         0.00    1.74 r
u8/Z (IVA)         0.43    2.17 f

```

report_timing
1014

u7/B (OR3)	0.00	2.17 f
u7/Z (OR3)	1.24	3.41 f
z2 (out)	0.00	3.41 f
data arrival time		3.41
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-3.41

slack (VIOLATED)		-3.41

The following example shows a summary of the total slack between the startpoint and endpoint:

```
pt_shell> report_timing -path_type summary
```

```
*****
```

```
Report : timing
    -path_type summary
    -delay_type max
    -max_paths 1
    -sort_by slack
```

```
...
```

```
*****
```

Startpoint	Endpoint	Slack
BlkA/ff3/CK (SDFFPCRQX1H1T10W)	BlkB/ff3/D (SDFFPCRQX1H1T10W)	9.45

The following example reports transition time and capacitance:

```
pt_shell> report_timing -transition_time -capacitance
```

```
*****
```

```
Report : timing
    -path_type full
    -delay max
    -max_paths 1
    -transition_time
    -capacitance
```

```
Design : counter
```

```
...
```

```
*****
```

```
Startpoint: ffa (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	Cap	Trans	Incr	Path
clock CLK (rise edge)			0.00	0.00
clock network delay (ideal)			0.00	0.00
ffa/CLK (DTC10)	0.00	0.00	0.00	0.00 r

ffa/Q (DTC10)	3.85	0.57	1.70	1.70 f
U7/Y (IV110)	6.59	1.32	0.84	2.55 r
U12/Y (NA310)	8.87	2.47	2.04	4.58 f
U17/Y (NA211)	4.87	1.01	1.35	5.94 f
U23/Y (IV120)	2.59	0.51	0.37	6.30 r
U15/Y (BF003)	2.61	0.88	0.82	7.12 f
U16/Y (BF003)	2.61	1.46	0.99	8.11 r
U10/Y (AN220)	2.63	0.46	1.04	9.15 r
ffd/D (DTN10)		0.46	0.00	9.15 r
data arrival time				9.15
clock CLK (rise edge)			10.00	10.00
clock network delay (ideal)			0.00	10.00
ffd/CLK (DTN10)				10.00 r
library setup time			-1.33	8.67
data required time				8.67

data required time				8.67
data arrival time				-9.15

slack (VIOLATED)				-0.48

The following example includes nets and input pins and reports transition time and capacitance:

```
pt_shell> report_timing -transition_time -capacitance -nets -input_pins
```

```
*****
```

```
Report : timing
```

```
  -path_type full
  -delay max
  -input_pins
  -nets
  -max_paths 1
  -transition_time
  -capacitance
```

```
Design : counter
```

```
...
```

```
*****
```

Startpoint: ffa (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	Fanout	Cap	Trans	Incr	Path
<hr/>					
clock CLK (rise edge)				0.00	0.00
clock network delay (ideal)				0.00	0.00
ffa/CLK (DTC10)			0.00	0.00	0.00 r
ffa/Q (DTC10)			0.57	1.70	1.70 f
b (net)	2	3.85			
U7/A (IV110)			0.57	0.00	1.70 f
U7/Y (IV110)			1.32	0.84	2.55 r
QA (net)	5	6.59			
U12/A (NA310)			1.32	0.00	2.55 r

report_timing

1016

U12/Y (NA310)			2.47	2.04	4.58 f
n9 (net)	3	8.87			
U17/A (NA211)			2.47	0.00	4.58 f
U17/Y (NA211)			1.01	1.35	5.94 f
n14 (net)	2	4.87			
U23/A (IV120)			1.01	0.00	5.94 f
U23/Y (IV120)			0.51	0.37	6.30 r
n13 (net)	1	2.59			
U15/A2 (BF003)			0.51	0.00	6.31 r
U15/Y (BF003)			0.88	0.81	7.12 f
n12 (net)	1	2.61			
U16/B2 (BF003)			0.88	0.00	7.12 f
U16/Y (BF003)			1.46	0.99	8.11 r
n7 (net)	1	2.61			
U10/A (AN220)			1.46	0.00	8.11 r
U10/Y (AN220)			0.46	1.04	9.15 r
n15 (net)	1	2.63			
ffd/D (DTN10)			0.46	0.00	9.15 r
data arrival time					9.15

clock CLK (rise edge)			10.00	10.00	
clock network delay (ideal)			0.00	10.00	
ffd/CLK (DTN10)					10.00 r
library setup time			-1.33	8.67	
data required time					8.67

data required time					8.67
data arrival time					-9.15

slack (VIOLATED)					-0.48

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 Path	Fanout	Cap	DTrans	Trans	Delta	Incr
---------------	--------	-----	--------	-------	-------	------

```

-----
-----  

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 an unconstrained path. It should be noted that for unconstrained paths, PrimeTime removes any clock network delay and external delay from the path report. However, the paths are sorted by the actual arrival time including clock network delay and external delay. In this example, the second path reported has greater delay than the first since the clock network delay used to find the first path was greater.

```

pt_shell> report_timing -to an2/B -delay_type max_rise -path_type full_clock \
           -nworst 2
*****
Report : timing
  -path_type full_clock
  -delay max_rise
  -nworst 2
  -max_paths 2
...
*****
Startpoint: ff1 (rising edge-triggered flip-flop clocked by SYSCLK2)
Endpoint: an2/B (internal pin)
Path Group: (none)
Path Type: max

Point                         Incr      Path
-----
ff1/CP (FD2)                  0.00     0.00 r
ff1/Q (FD2)                   1.34     1.34 r
an1/Z (AN2)                   0.48     1.82 r
an2/B (sub) <-                0.00     1.82 r
data arrival time              1.82
-----
(Path is unconstrained)

Startpoint: ff0a (rising edge-triggered flip-flop clocked by SYSCLK1)
Endpoint: an2/B (internal pin)
Path Group: (none)
Path Type: max

Point                         Incr      Path
-----
ff0a/CP (FD2)                 0.00     0.00 r
ff0a/Q (FD2)                  1.34     1.34 r
an0/Z (AN2)                   0.91     2.25 r
an1/Z (AN2)                   0.48     2.73 r
an2/B (sub) <-                0.00     2.73 r
data arrival time              2.73
-----
(Path is unconstrained)

```

The following example shows the **-full_clock_expanded** option. In this example the DC2 clock is a generated clock from the DC1 clock, which is a generated clock from CLK. When the **-full_clock_expanded** option is used, the path between the primary CLK clock and the generated DC2 clock is shown going through register levels and through other generated clocks. This helps in debugging clock source latency values.

```

pt_shell> report_timing -path_type full_clock_expanded -group DC2
*****
Report : timing
  -path_type full_clock_expanded
  -delay max
  -max_paths 1
  -group DC2

```

...

```
*****
```

Startpoint: ff3 (rising edge-triggered flip-flop clocked by DC2)
Endpoint: ff3 (rising edge-triggered flip-flop clocked by DC2)
Path Group: DC2
Path Type: max

Point	Incr	Path
<hr/>		
clock DC2 (rise edge)	0.00	0.00
clock CLK (source latency)	163.00	163.00
clk (in)	0.00	163.00 r
u1/Y (inv1a4) (gclock source)	83.32	246.32 f
u1/Y (inv1a4)	0.00	246.32 f
u2/Y (buf1a4)	340.72	587.04 f
u3/Y (inv1a2)	95.79	682.83 r
u4/A (buf1a2) (gclock source)	0.00	682.83 r
u4/Y (buf1a2)	324.15	1006.98 r
ff3/CLK (fd1a1)	0.00	1006.98 r
ff3/CLK (fd1a1)	0.00	1006.98 r
ff3/Q (fd1a1)	939.28	1946.26 f
u10/Y (or4b1)	990.35	2936.62 f
ff3/D (fd1a1)	0.00	2936.62 f
data arrival time		2936.62

(The required path not shown)

The following example shows a timing report with the **-derate** option specified:

```
pt_shell> report_timing -derate -from ff1/CP -to ff2/D -delay_type min
*****
```

Report : timing
-path_type full_clock_expanded
-delay min
-input_pins
-max_paths 1

```
...
```

```
*****
```

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: min

Point	Derate	Incr	Path
<hr/>			
clock clk (rise edge)	0.0000	0.0000	
clock source latency	0.0000	0.0000	
clk (in)	0.0000	0.0000 r	
u1/A (IBUF1)	0.8000	0.0000	0.0000 r
u1/Z (IBUF1)	0.8000	0.6814	0.6814 r
ff1/CP (FD1)	0.8000	0.0000	0.6814 r
ff1/CP (FD1)	0.8000	0.0000	0.6814 r
ff1/Q (FD1)	1.0000	1.2927	1.9740 r

u3/A (IV)	1.0000	0.0000	1.9740	r
u3/Z (IV)	1.0000	0.2319	2.2059	f
ff2/D (FD1)	1.0000	0.0000	2.2059	f
data arrival time			2.2059	
clock clk (rise edge)	0.0000	0.0000		
clock source latency	0.0000	0.0000		
clk (in)	0.0000	0.0000	0.0000	r
u1/A (IBUF1)	1.2000	0.0000	0.0000	r
u1/Z (IBUF1)	1.2000	1.0221	1.0221	r
u2/A (IBUF1)	1.2000	0.0000	1.0221	r
u2/Z (IBUF1)	1.2000	0.8167	1.8388	r
ff2/CP (FD1)	1.2000	0.0000	1.8388	r
clock reconvergence pessimism	-0.3407	1.4981		
library hold time	1.0000	0.4000	1.8981	
data required time			1.8981	

data required time			1.8981	
data arrival time			-2.2059	

slack (MET)			0.3078	

Derate Summary Report

total derate : required time	-0.3065
total derate : arrival time	0.1703

total derate : slack	0.4768
slack (with derating applied) (MET)	0.3078
clock reconvergence pessimism (due to derating)	-0.3407

slack (with no derating) (MET)	0.4440

1

The following example shows a timing report where dynamic clock latency has been specified using the **set_clock_latency** command:

```
*****
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

The following example shows a timing report with the **-exceptions all** option specified:

```
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                  0.00      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

If you specify the **-pba_mode exhaustive** option, 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 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 -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)
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)
```

report_timing_derate

Reports derate factors annotated on the current design.

SYNTAX

```
int report_timing_derate
[-include_inherited]
[-variation]
[-aocvm_guardband]
[-pocvm_guardband]
[-pocvm_coefficient_scale_factor]
[-increment]
[-significant_digits digits]
[-nosplit]
[object_list]
```

Data Types

<i>digits</i>	int
<i>object_list</i>	list

ARGUMENTS

-include_inherited

Indicates that the derate factors reported on each object should also include those set by other objects in the design.

-variation

Indicates that only variation derate factors should be reported. If neither the *-variation* nor *-aocvm_guardband* option is specified, only deterministic derate factors are reported. The *-variation* and *-aocvm_guardband* options are mutually exclusive.

-aocvm_guardband

Indicates that only guardband derate factors should be reported. If neither the *-variation* nor *-aocvm_guardband* option is specified, only deterministic derate factors are reported. The *-variation* and *-aocvm_guardband* options are mutually exclusive.

-pocvm_guardband

Indicates that only guardband derate factors should be reported, in POCV mode. If neither the *-variation* nor *-pocvm_guardband* option is specified, only deterministic derate factors are reported. The *-variation* and *-pocvm_guardband* options are mutually exclusive in POCV mode.

-pocvm_coefficient_scale_factor

Indicates that only coefficient scale factors should be reported, in POCV mode.

-increment

Indicates that incremental derate factors should be reported. The *-increment* and *-aocvm_guardband* options are mutually exclusive.

-significant_digits digits

Specifies the number of digits after the decimal point to be displayed for values in the report. Allowed values are 0-13; 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 a list of cells, lib_cells, nets in the current design.

DESCRIPTION

Reports the derate factors either on the design, specific instances (cells or library cells) contained in the design or on specific nets contained in the design. If this command is successful a Boolean of true or 1 is returned; otherwise, it returns a false or 0.

If this command is called with no arguments, a separate report is invoked for the derate factors stored on the top design (hereafter referred to as global derate factors) and also for each instance and net that has derate factors set on it.

If this command is called with the **-include_inherited** Boolean argument, the derate report for each instance also contains derate factors set from other objects in the design that were inherited by the instance being reported. Examples of this would be a leaf cell inheriting cell derate factors set on the hierarchical cell to which it belongs, or a leaf cell inheriting a derate factor set on the library cell of which it is an instantiation. In addition to the above, each row containing an inherited derate factor that also contains a column entry indicating the object from which it was inherited. The **-include_inherited** argument does not have any effect on derate reports for nets.

If the command is called with the **object_list** option specified, derate reports are only issued for the instances or nets specified in the object list. If no derate factors exist on any of the objects specified, a warning message is issued for that object. If the **-include_inherited** Boolean argument is also specified, inherited derate factors are reported as outlined in the above paragraph.

EXAMPLES

In the following example derate factors have only been set globally (on the design), therefore only global derate factors are reported.

```
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
```

```
*****
Report : timing derate
Design : simple_path
*****
```

	Clock				Data			
	Rise		Fall		Rise			
Fall	Early	Late	Early	Late	Early	Late	Ea	
rly	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	--	--	--	--	--	--	-	
- --								
1								

In the following example incremental derate factors have only been set globally (on the design), therefore only global incremental derate factors are reported.

```
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
*****
```

```
Report : timing derate
    -scalar -increment
Design : simple_path
*****
```

	Clock				Data			
	Rise		Fall		Rise			
Fall	Early	Late	Early	Late	Early	Late	Ea	
rly	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	--	--	--	--	--	--	-	

```
report_timing_derate
1026
```

- - -

1

In the following example derate 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/u5' are reported.

```
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_cell H1]
pt_shell> set_timing_derate -late 1.12 [get_cell H1]
pt_shell> report_timing_derate -include_inherited [get_cells H1/u5]
*****
```

```
Report : timing derate  
          -include_inherited
```

Design : top

SEE ALSO

```
set_timing_derate(2)
reset_timing_derate(2)
timing derate report compatibility(3)
```

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

<i>from_list</i>	list
<i>through_list</i>	list
<i>to_list</i>	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

<i>from_list</i>	list
<i>through_list</i>	list
<i>to_list</i>	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.

DESCRIPTION

The **report_units** command displays the units used by the current design. To generate the report, the design should be loaded and linked to a library. The units are always reported in reference to the MKS units such as Farad, Amp, Ohm, Second, and Volt. The resistance unit is inferred from the time and capacitance units. Both time and capacitance units are required for PrimeTime. If either of them is missing, the units are undefined and is reported as "Not specified". The most common cause of this problem occurs when the library is in an old format that needs to be recompiled with the latest Library Compiler.

If power analysis is on, the power units is also reported.

EXAMPLES

The following report is an example of the unit information that is reported:

```
pt_shell> report_units
*****
Report : units
Design : simple
Version: D-2010.06
Date   : Mon Jun 28 11:55:27 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
```

SEE ALSO

```
read_db(2)
set_min_library(2)
```

report_vcd_hierarchy

Reports the hierarchy in the VCD file.

SYNTAX

```
string report_vcd_hierarchy
[-pipe_exec shell_command_string]
[-full_path]
[-find pattern_string]
[file_name]
```

Data Types

<i>shell_command_string</i>	string
<i>pattern_string</i>	string
<i>file_name</i>	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

Displays the indented hierarchy in the specified VCD file.

To read in a gzipped VCD file, a compressed VCD file, a VCD+ file, and a FSDB file, user's PATH should 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_waveform_integrity_analysis

Reports static or dynamic constraints for waveform integrity analysis.

SYNTAX

```
int report_waveform_integrity_analysis
[-static]
[-dynamic]
[-nosplit]
[-significant_digits num_digits]
[object_list]
```

Data Types

<i>num_digits</i>	integer
<i>object_list</i>	list

ARGUMENTS

-static	Reports the static constraint.
-dynamic	Reports the dynamic constraint.
-nosplit	Prevents line splitting in the report when columns overflow.
-significant_digits <i>num_digits</i>	Specifies the number of digits displayed in the report.
object_list	Specifies a list of cell input pins.

DESCRIPTION

This command reports the constraint limit for waveform integrity analysis set by the **set_waveform_integrity_analysis** command.

If the list of cell input pins are provided, the constraint limit on the specified cell inputs is reported. If there is no list of cell input pins, all constraints of both design-level and pin-level are reported.

EXAMPLES

The following example shows static constraints for waveform integrity analysis.

```
pt_shell> report_waveform_integrity_analysis -static
```

```
*****
```

```
Report : waveform integrity tolerance
         -static
Design : test_case
*****
```

Cell pins	Tolerance
DFF1/CLK	0.3000
DFF1/D	0.3000
DFF2/CLK	0.3000
DFF2/D	0.3000

SEE ALSO

```
remove_waveform_integrity_analysis(2)
report_constraint(2)
set_waveform_integrity_analysis(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
cell1	inv_top tst	WLM1 WLM2	my_lib	user-specified 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_aocvm_table_group

The **reset_aocvm_table_group** command removes the association between a named AOCV table set and a hierarchical instance or between a library cell and a AOCV derating group.

SYNTAX

```
status reset_aocvm_table_group
[object_list]
```

Data Types

object_list list

ARGUMENTS

object_list
 Specifies the list of design objects.

DESCRIPTION

The **reset_aocvm_table_group** command removes the association between a named AOCV table set and a hierarchical instance or between a library cell and a AOCV derating group, where such an association was previously specified with the **set_aocvm_table_group** command.

EXAMPLES

The following example specifies an association between the hierarchical cell H1 and the AOCV table set A1 and then removes this association. After the association is removed, cells (or nets) fully enclosed within H1 derive their AOCV derating from a higher level hierarchical cell that fully encloses H1 and has a named AOCV table set associated with it. If no such hierarchical cell is found, the unnamed AOCV table set is used.

```
pt_shell> set_aocvm_table_group A1 [get_cells H1]
```

```
pt_shell> reset_aocvm_table_group [get_cells H1]
```

SEE ALSO

`get_timing_paths(2)`
`read_aocvm(2)`
`remove_aocvm(2)`
`report_aocvm(2)`
`report_timing(2)`
`set_aocvm_coefficient(2)`
`set_aocvm_table_group(2)`
`write_binary_aocvm(2)`

reset_design

Removes user specified information from design.

SYNTAX

```
string reset_design
[-timing]
```

ARGUMENTS

-timing
Resets the timing information on the current design.

DESCRIPTION

Removes all attributes from the design. It does not matter whether these attributes are user-defined, command implemented, or read from the design database. Attributes include, but are not limited to, exceptions, input/output delays, and clocks.

EXAMPLES

The following command resets the attributes on the current design.

```
pt_shell> reset_design
```

SEE ALSO

```
current_design(2)
remove_user_attribute(2)
```

reset_eco_options

Resets options specified by the **set_eco_options** command.

SYNTAX

```
status reset_eco_options
```

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **reset_eco_options** command removes all the ECO options specified by the **set_eco_options** command.

Distributed Multi-Scenario Analysis

To execute the **reset_eco_options** command in all scenarios, use the **remote_execute** command. For example:

```
remote_execute -verbose {  
    reset_eco_options  
}
```

Note: You cannot execute the **reset_eco_options** command in the master process.

EXAMPLES

The following example shows how to use the **reset_eco_options** command.

```
pt_shell> reset_eco_options
```

SEE ALSO

```
report_eco_options(2)  
set_eco_options(2)
```

reset_mode

Resets cell mode groups or design mode groups to the default state.

SYNTAX

```
Boolean reset_mode
[-type cell | design]
[-group group_list]
[instance_list]
```

Data Types

<i>group_list</i>	list
<i>instance_list</i>	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_multi_input_switching_coefficient

Resets user-specified multi-input switching (MIS) coefficients.

SYNTAX

```
int reset_multi_input_switching_coefficient
    -all | object_list
```

Data Types

object_list list

ARGUMENTS

-all

Resets all the library cells specified for multi-input switching analysis.
You cannot use this option together with the *object_list* option.

object_list

Specifies a list of library cells to be reset. You cannot use this option together with the **-all** option.

DESCRIPTION

This command resets the user-specified base coefficients for multi-input switching (MIS) analysis. If you reset the coefficient of a library cell, then the tool does not perform multi-input switching analysis on that library cell.

EXAMPLES

The following example resets the base coefficients of library cell AND2 in the library MY_LIB:

```
pt_shell> reset_multi_input_switching_coefficient [get_lib_cells MY_LIB/AND2]
```

SEE ALSO

```
set_multi_input_switching_coefficient(2)
report_multi_input_switching_coefficient(2)
si_enable_multi_input_switching_analysis(3)
si_enable_multi_input_switching_timing_window_filter(3)
```

reset_noise_parameters

Resets the noise analysis parameters for the current design.

SYNTAX

```
int reset_noise_parameters
```

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_ocvm_table_group

Resets the association between an AOCV or POCV named table group or a based POCV or AOCV derating group and a list of design objects, where an association has been performed earlier using the **set_ocvm_table_group** command.

SYNTAX

```
status reset_ocvm_table_group
-type aocvm | pocvm
[-table_type coefficient | distance]
[object_list]
```

Data Types

object_list list

ARGUMENTS

```
-type aocvm | pocvm
    Specifies AOCV or POCV tables.

-table_type coefficient | distance
    Specifies coefficient or distance-based POCV tables. The default is
    coefficient. Do not use this option for AOCV.

object_list
    Specifies a list of design objects.
```

DESCRIPTION

The **reset_ocvm_table_group** command removes the association between a named AOCV or POCV table set and a hierarchical instance or between a lib_cell and a AOCV or POCV derating group, where such an association has been specified previously using the **set_ocvm_table_group** command.

EXAMPLES

In the following example, the **set_ocvm_table_group** command specifies an association between the hierarchical cell H1 and the named POCV coefficient table set A1. The **reset_ocvm_table_group** command that follows removes this association. Cells (or nets) fully enclosed within H1 then derive their POCV coefficients from a higher-level hierarchical cell that fully encloses H1 and has a named POCV coefficient table set associated with it. If no such hierarchical cell is found, the unnamed POCV coefficient table set is used.

```
pt_shell> set_ocvm_table_group -type pocvm -table_type coefficient A1 [get_cells H1]
pt_shell> reset_ocvm_table_group -type pocvm -table_type coefficient [get_cells H1]
```

SEE ALSO

`get_timing_paths(2)`
`report_ocvm(2)`
`report_timing(2)`
`set_ocvm_table_group(2)`

reset_path

Resets specified paths to single-cycle behavior.

SYNTAX

```
Boolean 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

<i>from_list</i>	list
<i>rise_from_list</i>	list
<i>fall_from_list</i>	list
<i>through_list</i>	list
<i>rise_through_list</i>	list
<i>fall_through_list</i>	list
<i>to_list</i>	list
<i>rise_to_list</i>	list
<i>fall_to_list</i>	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.

-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, as long as 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]
object_list
```

Data Types

<i>group_name_list</i>	list
<i>object_list</i>	list

ARGUMENTS

-groups *group_name_list*
Specifies the power group or groups in which the derate factors of all the cells 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

```
int reset_rtl_to_gate_name
```

ARGUMENTS

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 scaling that was done previously using the **scale_parasitics** command.

SYNTAX

```
int reset_scale_parasitics
[net_list]
```

Data Types

net_list list

ARGUMENTS

net_list

Limits the reset of scaling to this list of nets. If this option is used, it is assumed that net-specific scaling has been done for those nets. If not, an error is generated.

DESCRIPTION

Use the **reset_scale_parasitics** command to reset the scaling of parasitics that were previously done through the **scale_parasitics** command.

EXAMPLES

This example resets scaling of parasitics previously done.

```
pt_shell> reset_scale_parasitics
```

SEE ALSO

```
read_parasitics(2)
scale_parasitics(2)
report_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

<i>state</i>	string
<i>name_list</i>	string
<i>object_list</i>	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:

reset_switching_activity

1060

```
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 user-specified derate factors set either on a design or on a specified list of instances, such as cells, nets or library cells.

SYNTAX

```
int 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
Indicates that net derate factors within specified hierarchical cells should be reset.

-scalar
Indicates that only derate factors for deterministic delays should be reset.

-variation
Indicates that only derate factors for statistical delays should be reset.

-aocvm_guardband
Indicates that only AOCV guardband derate factors should be reset.

-pocvm_guardband
Indicates that only POCV guardband derate factors should be reset.

-pocvm_coefficient_scale_factor
Indicates that only POCV coefficient scale factors should be reset.

-increment
Indicates that only incremental derate factors for the specified (scalar/variation) delays should be reset.

object_list
Specifies a list containing the design, cells, library cells, or nets which are reset.

DESCRIPTION

Call this command with no arguments to reset all derate factors set on the design (to the default value, 1.0). This includes both derate factors set globally on the design and those set on specific instances in the design. Call this command with a list of objects to reset a specific set of objects.

The *object_list* option may be used to reset derate factors on specific objects (cells, library cells, or nets) in the design. All derate factors are reset on each instance. If the *object_list* option contains a design, then only global derate factors are reset and instance-specific derate factors are not reset.

Derates must be reset using the same object type used to apply the original derate. For example, global derates can not be reset on a library cell or cell instance basis. However, derates of 1.0 can be set on more specific object types to effectively remove the effects of derating.

Heterogeneous collections of objects may be combined and passed to this command.

If neither the *-scalar* nor *-variation* options are specified, then both deterministic and variation derates are reset.

In advanced OCV (AOCV) analysis, the increment derate factors are reset when the *-increment* option is specified OR when neither *-scalar*, *-variation*, *-aocvm_guardband* or *-increment* is specified. The *-incrementf* option cannot be specified with the *-aocvm_guardband* option.

In parametric OCV (POCV) analysis, the increment derate factors are reset when the *-increment* option is specified OR when neither *-scalar*, *-variation*, *-pocvm_guardband* or *-increment* is specified. The *-incrementf* option cannot be specified with the *-pocvm_guardband* option.

If the *object_list* contains any hierarchical cells, then all cells within that hierarchical cell and also all cells of all lower hierarchies will have their inherited derate factors updated accordingly. In addition, if the *-hierarchical_net_delay* option is specified, each net and every net in each hierarchical cell in the *object_list* option will have their inherited derate factors updated. Note that derate factors set specifically on an object are not overwritten by a set/reset command on its parent hierarchical cell.

If a portion of a hierarchical net is reset, then all portions of that hierarchical net are also reset and a warning is issued.

EXAMPLES

The following command resets all (deterministic, statistical, incremental, non-incremental) derate factors set globally on the design and those set on specific instances in the design.

```
pt_shell> reset_timing_derate
```

The following command resets both deterministic and statistical incremental derate factors set globally on the design and those set on specific instances in the design.

```
pt_shell> reset_timing_derate -increment
```

The following command resets the derate factors set globally on the design MY DESIGN only.

```
pt_shell> reset_timing_derate [get_design MY DESIGN]
```

The following example resets the derate factors set on cell U1.

```
pt_shell> reset_timing_derate [get_cell U1]
```

The following example resets the derate factors set on net w1.

```
pt_shell> reset_timing_derate [get_net w1]
```

The following command resets the derate factors set on all cells matching "*" and on all nets matching "w*".

```
pt_shell> reset_timing_derate [add_to_collection [get_cells *]  
[get_nets w*] ]
```

The following example resets the derate factors set on all instances of library cell IV in the library MY_LIB.

```
pt_shell> reset_timing_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 and as a result U1 inherits the derate factors that are set on H1.

```
pt_shell> reset_timing_derate [get_cells H1/U1]
```

Assuming that the hierarchical cell H1 also contains a hierarchical cell H2, then the following command resets the derate factors set on H2 and as a result H2 and all of its cells inherit the derate factors that are set on H1.

```
pt_shell> reset_timing_derate [get_cells H1/H2]
```

Assuming that the hierarchical cell H1 also contains a hierarchical cell H2, then the following command resets the derate factors set on H2 and as a result H2 and all of its cells AND NETS inherit the derate factors that are set on 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)
```

restore_session

Restore a PrimeTime session from a directory saved by the **save_session** command.

SYNTAX

```
int restore_session  
[-name session_name]  
dir_name
```

Data Types

<i>session_name</i>	string
<i>dir_name</i>	string

ARGUMENTS

-name session_name

This option, if specified when restoring a timing path session, allows assigning a name for the restored path session.

dir_name

Specifies the name of a directory to read the session information from.

DESCRIPTION

Use this command to read a directory that was written by the **save_session** command and restore a PrimeTime session to the same state as the session where the **save_session** command was issued.

Ensure the version of PrimeTime used to restore a session with the **restore_session** command is the same version used when saving the session with the **save_session** command. Locate the version used to save the session from the README file located in the session directory.

No current designs or libraries can be loaded to execute the **restore_session** command. Either issue this command in a clean shell or use the **remove_designs -all** and **remove_library -all** commands to clear the current session before restoring a new one.

The command first tries to grab all licenses that were present at the time of the **save_session** command. If this fails, restoring a session is aborted.

The design data is restored from data in the session directory. The **restore_session** command reads the library files that are needed to restore the session.

The restore directory contains the lib_map ASCII file name. This file contains the path and leaf name for each technology library needed to restore the session. If necessary, you can edit the path names to give the **restore_session** command the updated locations of the needed technology library files. The leaf names of those files cannot be changed. The technology libraries are assumed to be the same as those used when the **save_session** command was issued.

When a saved image is restored, any variable that existed in the saved image are restored to its saved value. However, any variables that exist in an analysis before the **restore_session** command is issued, which were not present in the saved image, remain unchanged.

An arbitrary collection of timing_path objects generated in a PrimeTime session can be saved using the **save_session** command with the option *-only_timing_paths*. In that case, only the timing_path collection and supporting data are saved and not the full design or timing information. Subsequently, multiple path sessions can be loaded into memory in a regular PrimeTime environment using the **restore_session** command. If a design has been instantiated prior to restore of a path session, then an error is generated and the path collection is not loaded. By default the name of the path collection restored is the same as the name of the session directory and each timing_path object within the restored collection inherits the path session name as a read-only application attribute *session_name*. The default path session name can be overwritten by using the *-name* option.

EXAMPLES

This example reads a PrimeTime savable image in a directory named state1.

```
pt_shell> restore_session state1
```

The following example restores a collection of timing paths:

```
pt_shell> restore_session path_session_dir -name mypaths
```

SEE ALSO

[save_session\(2\)](#)

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 *design_name.lib.db*. If you use the **-output** option, the tool writes the db file to *output_file.lib.db* file.

-library_cell

Specifies the model should be written as a library cell rather than a wrapper design and core library cell. 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 *design_name_core*.

If you use this option, no wrapper design is written, and the model in the *output_file.lib.db* file is named *design_name*.

DESCRIPTION

This command saves the quick timing model and indicates to PrimeTime that the model being defined is completed. All QTM commands must be issued 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 in db format.

```
pt_shell> save_qtm_model -format db
```

save_qtm_model

1068

The following example saves a library cell quick timing model 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\(2\)](#)

save_session

Saves the data of a PrimeTime session in the named directory so that you can later restore the session with the **restore_session** command.

SYNTAX

```
int save_session
[-include incl_list]
[-only_used_libraries]
[-only_timing_paths path_collection]
[-name session_name]
[-exclude HyperScale_data | full_analysis_data]
[-disable_common_data_sharing]
[dir_name]
```

Data Types

<i>incl_list</i>	list
<i>path_collection</i>	collection
<i>session_name</i>	string
<i>dir_name</i>	string

ARGUMENTS

-include incl_list

Specifies additional data to be included by the **save_session** command. Allowed values are

- **libraries** - Includes the loaded libraries to the saved session. After you restore an image that was saved with the **-include libraries** option, the current session relies on the saved image for its library data, imposing the following restrictions:

- PrimeTime updates the **search_path** variable to include the library location within the saved image.

- A subsequent **save_session** command cannot overwrite the saved image containing the libraries of the present session.

- Any subsequent **save_session** command also includes the libraries.

- **gca_session** - Includes the GCA session to the saved session. This forcefully saves the GCA session. This session is used when running the PrimeTime GCA commands.

- **timing_paths** - Includes timing path collections to the saved session.

-only_used_libraries

By default, the **save_session** command saves references to all loaded logic libraries at the saving of the session. All of these libraries are required to restore the session later. This option saves references to only libraries that are in use, which are needed to restore the session later.

-disable_common_data_sharing
By default, a multi_scenario **save_session**, whether issued at the master or at the slaves via **remote_execute**, will share common data between scenarios. This option disables the feature for the present save, and the resulting image will have no dependencies on common data.

-only_timing_paths path_collection
Saves the given timing path collection to disk in a representation that is fully independent of timing data and can be subsequently restored along with any number of other path sessions for analysis. If you save a session with this option, you can restore the session only when running interactive multi-scenario analysis (IMSA) in the GUI. You cannot use this option with the **-only_used_libraries** option.

-name session_name
Specifies a name for the saved path session when used together with the **-only_timing_paths** option.

-exclude HyperScale_data | full_analysis_data
In the HyperScale flow, the **save_session** command saves additional data related to the flow by default. This option excludes HyperScale data or the normal PrimeTime session data from the saved session. Allowed values are

- **HyperScale_data** - Excludes HyperScale data.
- **full_analysis_data** - Skips saving normal PrimeTime session.

dir_name
Specifies the directory to save the session to. This option is optional for the HyperScale flow. Otherwise, it is required. If you are using a flow other than the HyperScale flow, and the named directory does not exist, the **save_session** command attempts to create it. If it already exists, the **save_session** command attempts to delete the directory before recreating it. If PrimeTime is unable to delete the directory due to a lack of write permission, an error is issued and the session is not saved.

DESCRIPTION

Use this command to create a repository of data to save the current PrimeTime session to. It is required that you use the **dir_name** option to specify the name of the directory to save the session. If the directory does not exist, a new one is created and the data for the session is written into the directory.

If the directory target save directory already exists, the default behavior is to always overwrite data. Otherwise, a new directory is created to write the data.

The **save_session** command does not perform an implicit **update_timing**; therefore, you can save sessions before using the **update_timing** command. Only a linked design can be saved so the **save_session** might cause an implicit link to be performed. If there are incremental timing updates pending, PrimeTime performs an implicit **update_timing**, as needed. If noise is to be included, an implicit **update_noise** is also issued, as needed.

Note that high-capacity settings are saved and restored. Therefore, restoring a

session results in the saved session's high-capacity mode overriding the mode in the session where the **restore_session** command was issued. Additionally, in multicore analysis mode the distributed multicore analysis settings are saved in the multicore_compute_resources.tcl file located in the **save_session** directory. During a **restore_session** this script is sourced to launch the multicore slaves.

Note: The tool does not save collections that involve timing objects, Tcl procedures, threaded multicore settings, and warning and informational message suppressions. Also, if there are other designs in PrimeTime memory, other than the linked design, they are not saved. These designs must be separately read into the restored session.

Ensure the version of PrimeTime used to restore a session with the **restore_session** command is the same version used when saving the session with the **save_session** command. Locate the version used to save the session from the README file located in the session directory.

An arbitrary collection of timing_path objects generated in a PrimeTime session can be saved using the **save_session** command with the **-only_timing_paths** option. In that case, only the timing_path collection and supporting data is saved and not the full design or timing information. Subsequently, multiple path sessions can be loaded into memory in a regular PrimeTime environment using the **restore_session** command.

When **save_session** is called in a distributed multi-scenario analysis (DMSA) run, certain data, such as parasitic and physical DB information, are stored in a common data directory named **common_data**. The images saved by each scenario contain softlinks to the data in this common data directory to avoid a duplication of storage of the data that scenarios share in common. When **save_session** is issued directly to the master session, the common data directory is created in the specified image directory. When **save_session** is issued to the slaves via **remote_execute**, the location of the common data directory is based on whether the image directory specified to the command is relative or absolute:

- For absolute paths, the common data is located in the parent directory of the specified directory. All scenario images should share the same parent directory to maximise sharing.
- For relative paths, the common data directory is located directly under the multi-scenario working directory. To execute a multi_scenario **save_session** with common data sharing disabled, use the **-disable_common_data_sharing** option.

EXAMPLES

The following example saves a PrimeTime session in the **state1** directory:

```
pt_shell> save_session state1
```

The following example generates and saves a collection of timing paths:

```
pt_shell> set paths [get_timing_paths -max 10 -nworst 10]
pt_shell> save_session -only_timing_paths $paths path_session_dir
```

SEE ALSO

`restore_session(2)`

scale_parasitics

Used to scale the parasitics in memory.

SYNTAX

```
int scale_parasitics
[-resistance_factor r_factor]
[-ground_capacitance_factor c_factor]
[-coupling_capacitance_factor cc_factor]
[net_list]
```

Data Types

<i>r_factor</i>	float
<i>c_factor</i>	float
<i>cc_factor</i>	float
<i>net_list</i>	list

ARGUMENTS

-resistance_factor *r_factor*
A positive floating point number that specifies the factor to apply for scaling resistances.

-ground_capacitance_factor *c_factor*
A positive floating point number that specifies the factor to apply for scaling ground capacitances.

-coupling_capacitance_factor *cc_factor*
A positive floating point number that specifies the factor to apply for scaling coupling capacitances.

net_list
Limits the scaling to this list of nets.

DESCRIPTION

This command enables scaling of resistance, grounded capacitance or coupling capacitance values. The primary purpose is to allow you to perform a single parasitic extraction run, then to scale parasitics for effects, such as resistance values for temperature dependence.

The command works for detailed parasitics as well as Pi models. The command works irrespective of the source of the parasitics information (whether SPEF, SBPF, DSPF or RSPF). Any subsequent calculations use the scaled parasitics instead of the original parasitics. Also, if the parasitics are written out using the **write_parasitics** command, scaled parasitics are written out.

Also, the parasitics can be scaled either globally or to a set of specific nets. Multiple scale parasitics are with respect to the original parasitics, and are not cumulative.

If conflicting coupling capacitance scale factors are given for two coupled nets, the last command takes precedence. When parasitics are read in again for previously scaled nets, the global scaling is still honored, but the net-specific scaling is ignored.

EXAMPLES

This example scales the ground capacitances by a factor of 1.3, resistance values by a factor 1.1 and the coupling capacitance values by a factor of 0.95.

```
pt_shell> scale_parasitics -resis 1.1 -ground 1.3 -coupling 0.95
```

This example scales the ground capacitances by a factor of 2.1, and the coupling capacitance values by a factor of 0.95 for one net n1.

```
pt_shell> scale_parasitics -ground 2.1 -coupling 0.95 [get_net n1]
```

SEE ALSO

```
read_parasitics (2)  
reset_scale_parasitics (2)  
report_scale_parasitics (2)
```

set_active_clocks

Sets a group of clocks to be active in the current analysis scope.

SYNTAX

```
Boolean 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* options are mutually exclusive.

```
all_clocks
```

Specifies to analyze all clocks simultaneously. The *active_clock_list* and *all_clocks* options 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* option.

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
[-rise]
[-fall]
[-min]
[-max]
[-from from_pins]
[-to to_pins]
[-clock rise | fall]
[-cond sdf_expression]
[-worst]
[-increment]
check_value
```

Data Types

<i>from_pins</i>	list
<i>to_pins</i>	list
<i>sdf_expression</i>	string
<i>check_value</i>	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 (**-from**) 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.

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
```

`set_annotated_check`

1080

...

```
*****
```

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)
```

Timing check type	Total	Annotated	NOT Annotated
cell min pulse width arcs	20	1	19

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

<i>internal_power</i>	float
<i>switching_power</i>	float
<i>leak_power</i>	float
<i>total_power</i>	float
<i>domain_clock</i>	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 or 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 are 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

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
<hr/>						
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%)	

Clock	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
<hr/>						
CKB	0.0000	2.000e-03	0.0000	2.000e-03		
CKA	1.000e-03	0.0000	0.0000	1.000e-03		
<hr/>						
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%)
<hr/>		
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 load_delay_type]
[-from from_pins]
[-to to_pins]
[-cond sdf_expression]
[-increment]
[-delta_only]
[-worst]
-variation variation_object
[-of_objects objects]
delay_value
```

Data Types

<i>load_delay_type</i>	string
<i>from_pins</i>	list
<i>to_pins</i>	list
<i>sdf_expression</i>	string
<i>objects</i>	list
<i>delay_value</i>	float

ARGUMENTS

-cell

Specifies that the delay annotated is a cell delay. The *-cell* and *-net* options are mutually exclusive; you must specify one, but not both.

-net

Specifies that the delay annotated is a net delay. The *-net* and *-cell* options are mutually exclusive; you must specify one, but not both.

-rise

Indicates 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

Indicates 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

Use this option only if the design is in min_max mode (min and max operating conditions). Specifies the minimum delay 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 delay for both data rise and data fall transitions.

-load_delay load_delay_type

Specifies whether load delay is to be included as part of annotated net delays or as part of annotated cell delays. Allowed values are *net* or *cell*. 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 *load_delay_type*.

-from from_list

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_list

Specifies a list of leaf cell pins and/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.

-of_objects objects_list

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.

-cond sdf_expression

Use this option only if the library has a condition attached to the specified delay timing arc; otherwise, an error message is generated. 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).

-increment

Specifies that the delay is to be incremented to the current delay of the specified timing arc. It should be noted that if this command is used before a timing update, the current delay could be 0.

-delta_only

Specifies that the annotated delay is to be added to the net delay value calculated by PrimeTime. You cannot use this option with the *-cell* option.

-worst

This option is not yet implemented, so it is ignored.

delay_value

Specifies the delay value between pins on the same cell, in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, *delay_value* must be expressed in nanoseconds.

```
-variation variation_object
```

Specify a variation to annotate on all arcs between the from and to pins. The variation_object must be created using the create_variation command.

DESCRIPTION

Annotates the cell delay between two or more pins on a cell, or the net delay between two or more pins on the same net, in the **current design** command. With the -net option, pins in the *from_list* must be cell output or inout pins, and pins in the *to_list* must be cell input or inout pins.

With the -cell option, both the -from and -to options are required.

Pins in the *from_list* must be cell input or inout pins, and pins in the *to_list* must be cell output or inout pins. 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 once 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_power

Annotate power on unresolved black-box cells or leaf cells.

SYNTAX

```
int set_annotated_power
-internal_power int_value
-leakage_power leak_value
[-rails rail_list]
cell_list
```

Data Types

<i>internal_power</i>	float
<i>leakage_power</i>	float
<i>rail_list</i>	list
<i>cell_list</i>	list

ARGUMENTS

- internal_power *int_value*
Specifies the internal power to be annotated.
- leakage_power *leak_value*
Specifies the leakage power to be annotated.
- [-rails *rail_list*]
Specifies a list of rails/power supply nets on which power is annotated.
- cell_list*
Specifies a list of cells on which power is annotated.

DESCRIPTION

The **set_annotated_power** command annotates the internal and leakage power in the unit of Watt on the given cells. 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, 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.

If the rail option is specified, the annotated power values will be applied to each power supply net (or rail in non-UPF mode) in the list. This feature is only valid when the variable `power_enable_multi_rail_analysis` is set to true.

Switching power (power of the nets connected to the output pins of the cell) cannot be annotated. Instead, one can set switching activities on the nets, thus the switching power can be calculated.

The annotated power is average power, so it has no effect on power waveform.

EXAMPLES

The following example shows how power is annotated, removed and reported.

```
pt_shell> set_annotated_power -int 1 -leak 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)
```

| Cell type                 | Total | Annotated | NOT 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

```

| Cell type                 | Total | Annotated | NOT Annotated |
|---------------------------|-------|-----------|---------------|
| unresolved black-box cell | 2     | 1         | 1             |
| leaf cell                 | 3     | 0         | 3             |
|                           | 5     | 1         | 4             |

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**

|                   |       |
|-------------------|-------|
| <i>slew_value</i> | float |
| <i>pin_list</i>   | 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\_aocvm\_coefficient**

Specifies the advanced on-chip variation (AOCV) coefficients on library cells, library timing arcs, and cells.

### **SYNTAX**

```
status set_aocvm_coefficient
 coefficient
 object_list
```

### **Data Types**

|                    |       |
|--------------------|-------|
| <i>coefficient</i> | float |
| <i>object_list</i> | list  |

### **ARGUMENTS**

|                    |                                                                                                        |
|--------------------|--------------------------------------------------------------------------------------------------------|
| <i>coefficient</i> | Specifies a coefficient value, which should be a floating-point value greater than zero.               |
| <i>object_list</i> | Specifies a list of library cells, library timing arcs, or leaf cells on which the coefficient is set. |

### **DESCRIPTION**

Sets AOCV coefficients on library cells, library timing arcs, and leaf cells. Coefficients are used to calculate cell path depths; the default depth increment for a cell arc in a path is 1, but you can specify another coefficient value with this command. AOCV coefficients are not required for an AOCV analysis; however, their use might increase the accuracy of the analysis.

The tool applies the coefficients in the following order of priority for cell arcs (in decreasing order of precedence):

- 1) Cell
- 2) Library timing arc
- 3) Library cell
- 4) 1.0 (default)

To display AOCV coefficients, use the **report\_aocvm -coefficient** command. To remove AOCV coefficients from a cell, use the **remove\_aocvm -coefficient** or the **reset\_design** command.

### **EXAMPLES**

The following examples specify AOCV coefficients on library cells and library timing arcs.

```
pt_shell> set_aocvm_coefficient 1.2 [get_lib_cells lib/AN2]
pt_shell> set_aocvm_coefficient 2.5 [get_lib_timing_arcs -from lib/AN2/A -to lib/
AN2/Z]
```

## SEE ALSO

```
read_aocvm(2)
remove_aocvm(2)
report_aocvm(2)
timing_aocvm_analysis_mode(3)
```

## **set\_aocvm\_table\_group**

Associates an AOCV named table group or Liberty-based AOCV derating group with the list of specified design objects.

### **SYNTAX**

```
status set_aocvm_table_group
aocvm_table_group_name
[object_list]
```

### **Data Types**

|                        |        |
|------------------------|--------|
| aocvm_table_group_name | string |
| object_list            | list   |

### **ARGUMENTS**

aocvm\_table\_group\_name  
Specifies the name of the AOCV table set.

object\_list  
Specifies the list of design objects.

### **DESCRIPTION**

The **set\_aocvm\_table\_group** command can operate on either hierarchical cells in the top-level design or on lib\_cells.

When operating on hierarchical cells in the top-level design, it sets the association between a named AOCV table group and a hierarchical instance. An AOCV table set is a collection of AOCV tables with the same name. Names are specified using the group\_name field in the AOCV table syntax.

When associations are specified between hierarchical cells in the design and named AOCV table sets, a cell (or net) derives its AOCV derating from the table set associated with the hierarchical cell or design that (a) fully contains the cell or net, and (b) is the lowest level (closest ancestor) hierarchical cell or design with an associated AOCV table set. If no such hierarchical cell or design containing the cell or net is found, the unnamed AOCV table set is used.

Note: A net inherits the AOCV derating set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating does not come from an AOCV table set that contains the block and fully encloses the net.

When operating on library\_cells the **set\_aocvm\_table\_group** command can be used to assign a Liberty-based AOCV table from a Liberty library to the given lib cell. The Liberty AOCV format allows for tables, which are not assigned to any library cell by default, to still be referenced by name. This command provides the ability to setup

this reference. The syntax for the P/AOCVM derate group to be referenced follows this pattern:

```
lib_name/[lib_cell_name]/derate_group_name
```

The lib\_cell\_name is optional and applicable only for lib\_cell based table. A lib\_cell based can only be assigned to a lib cell of the same name.

## EXAMPLES

The following example associates the hierarchical cell H1 with the AOCV table set A1.

```
pt_shell> set_aocvm_table_group A1 [get_cells H1]
```

```
pt_shell> sh cat test.aocvm
```

```
version: 3.0
group_name: A1
object_type: lib_cell
rf_type: rise
delay_type: cell
derate_type: late
object_spec: my_lib/*
voltage: 1.02
depth: 1 2 3
distance: 100 1000
table: 1.20 1.10 1.08
 1.22 1.15 1.11
```

## SEE ALSO

```
get_timing_paths(2)
read_aocvm(2)
remove_aocvm(2)
report_aocvm(2)
report_timing(2)
reset_aocvm_table_group(2)
set_aocvm_coefficient(2)
write_binary_aocvm(2)
```

## **set\_app\_var**

Sets the value of an application variable.

### **SYNTAX**

```
string set_app_var
 -default
 var
 value
```

### **Data Types**

|              |        |
|--------------|--------|
| <i>var</i>   | string |
| <i>value</i> | string |

### **ARGUMENTS**

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>-default</i> | Resets the variable to its default value.               |
| <i>var</i>      | Specifies the application variable to set.              |
| <i>value</i>    | 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_enable_page_mode 1
Error: "sh_enable_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 or 0, or is considered with a rising or falling transition.

### **SYNTAX**

```
string 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*, *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 pins, constant propagation is only executed forward. No backward constant propagation is performed.

### **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 or 0), or that only one type of transition (*rising* or *falling*) is considered.

Pins of a design may 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, this value is propagated through the network as long as 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.

Note: When a logic value is propagated onto a a net, the associated design rule checking (DRC\_ constraint checks aree 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 non-annotated 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 port *IN1* is at a constant logic value of 0.

```
pt_shell> set_case_analysis 0 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_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**

|                    |       |
|--------------------|-------|
| <i>setup_value</i> | float |
| <i>hold_value</i>  | float |
| <i>object_list</i> | 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**

|                       |        |
|-----------------------|--------|
| <i>clock_list</i>     | list   |
| <i>name</i>           | string |
| <i>comment_string</i> | string |

### **ARGUMENTS**

**-group** *clock\_list*

Specifies a list of clocks. **Note:** You can use the **-group** option more than one time in a single command execution.

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. Thus, 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.

Substitute the list you want for the *clock\_list* argument.

**-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, if CLK1 and CLK3 were also asynchronous with each other, you would need to specify an additional relationship between them.

## SEE ALSO

```
remove_clock_groups(2)
report_clock(2)
set_active_clocks(2)
set_false_path(2)
create_clock(2)
create_generated_clock(2)
```

## **set\_clock\_latency**

Specifies the latency of the clock network.

### **SYNTAX**

```
string set_clock_latency
[-clock clock_list]
[-rise]
[-fall]
[-min]
[-max]
[-source]
[-late]
[-early]
[-dynamic dynamic_component_of_delay]
[-pll_shift]
delay
object_list
```

### **Data Types**

|                                   |       |
|-----------------------------------|-------|
| <i>clock_list</i>                 | list  |
| <i>dynamic_component_of_delay</i> | float |
| <i>delay</i>                      | float |
| <i>object_list</i>                | 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.

```

-late
 Specifies clock late source latency.

-early
 Specifies clock early source latency.

-dynamic dynamic_component_of_delay
 Specifies dynamic component of clock latency value.

-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.

delay
 Specifies clock latency value.

object_list
 Lists clocks, ports, or pins.

```

## 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

set\_clock\_latency  
1110

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
```

## 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\_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**

|                   |       |
|-------------------|-------|
| <i>transition</i> | float |
| <i>clock_list</i> | 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
[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]
uncertainty
```

### **Data Types**

|                 |       |
|-----------------|-------|
| from_clock      | list  |
| rise_from_clock | list  |
| fall_from_clock | list  |
| rise_to_clock   | list  |
| fall_to_clock   | list  |
| to_clock        | list  |
| object_list     | list  |
| uncertainty     | float |

### **ARGUMENTS**

-from *from\_clock* -to *to\_clock*

These two options specify the source and 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\_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.

-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.

**-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.

## 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.

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_connection_class
connection_class_value
object_list
```

### **Data Types**

|                               |        |
|-------------------------------|--------|
| <i>connection_class_value</i> | string |
| <i>object_list</i>            | 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**

```
string set_context_margin
[-percent]
[-relax]
[-min]
[-max]
value
[object_list]
```

### **Data Types**

|             |        |
|-------------|--------|
| value       | double |
| object_list | list   |

### **ARGUMENTS**

|             |                                                          |
|-------------|----------------------------------------------------------|
| -percent    | Considered specified value as a percentage of the delay. |
| -relax      | Relaxes, instead of tightens, the constraint.            |
| -min        | Specifies the margin for min constraints.                |
| -max        | Specifies the margin for max constraints.                |
| value       | Determines the margin in absolute or percentage value.   |
| object_list | Indicates a list of cells or pins.                       |

### **DESCRIPTION**

The **set\_context\_margin** command specifies a margin to be added to or subtracted from input and output delay values when generated by the **write\_context** command. The margin can be specified as an absolute value or as a percentage of the constraint.

The constraints are first created using the **characterize\_context** or **create\_timing\_context** commands then written out using the **write\_context** command.

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 object list is specified. When determining the margin for a pin, the value set for the pin is used if specified. If you don't specify this value, the value set for the parent cell is used. If neither value is set, the value set for the current design is used. If no margin is specified, the default value of 0.0 is 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
```

The following command specifies that the max value input delays on I2/IN should be relaxed by 10 percent; that is, input delay values will be 10 percent less.

```
pt_shell> set_context_margin -max -percent 10.0 I2/IN
```

.

## SEE ALSO

`characterize_context(2)`  
`create_timing_context(2)`  
`report_context(2)`  
`write_context(2)`

## **set\_coupling\_separation**

Create a separation constraint on nets.

### **SYNTAX**

```
int set_coupling_separation
[-pairwise pnets]
nets
```

### **Data Types**

|              |      |
|--------------|------|
| <i>pnets</i> | list |
| <i>nets</i>  | list |

### **ARGUMENTS**

**-pairwise *pnets***

When **-pairwise *pnets*** is applied, all coupling capacitances between only *pnets* and *nets* are excluded.

**nets**

Lists the nets for which the separation constraint has to be applied. If this is not used with the **-pairwise** option, 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.

When **-pairwise *pnets*** is applied, all coupling capacitances only between *pnets* and *nets* are excluded. This is referred to as "pairwise exclusion."

Instances of this command are recorded for output by the **write\_changes** command. This feature allows the user to communicate the separation constraint to other implementation tools along with netlist changes.

The **set\_coupling\_separation** command returns a 1 if successful and a 0 if unsuccessful.

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)
write_changes(2)
set_si_delay_analysis(2)
set_si_noise_analysis(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**

```
int set_cross_voltage_domain_analysis_guardband
-early | -late
derate_value
object_list
```

### **Data Type**

|              |       |
|--------------|-------|
| 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.

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 guardband 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 PrimeTime's 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 guardband set for a supply net collection overwrites any previous settings for supply nets. Similarly, any new guardband set for a library cell collection overwrites any previous settings for library cells.

Please note that 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.

The following example sets a 10% delay degrataion 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\_current\_command\_mode**

### **SYNTAX**

```
string set_current_command_mode
-mode command_mode | -command command
stringcommand_mode
stringcommand
```

### **ARGUMENTS**

-mode *command\_mode*

Specifies the name of the new command mode to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

-command *command*

Specifies the name of the command whose associated command mode is to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

### **DESCRIPTION**

The **set\_current\_command\_mode** sets the current command mode. If there is a current mode in effect, the current mode is first canceled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

### **EXAMPLES**

The following example sets the current command mode to a mode called "mode\_1"

```
shell> set_current_command_mode -mode mode_1
```

The following example clears the current command mode without setting a new mode.

```
shell> set_current_command_mode -mode ""
```

The following example sets the current command mode to the mode associated with the command "modal\_command"

```
shell> set_current_command_mode -command modal_command
```

set\_current\_command\_mode  
1128

## **set\_current\_power\_domain**

Sets the specific power domains defined by the UPF **create\_power\_domain** command to be included in the power analysis.

### **SYNTAX**

```
status set_current_power_domain
 list_of_power_domains
```

### **Data Types**

```
list_of_power_domains list
```

### **ARGUMENTS**

*list\_of\_power\_domains*

Specifies the power domains defined in the design to be included in the power analysis. If *list\_of\_power\_domains* specifies an undefined power domain, the tool issues an error and does not change the current power domain setting.

### **DESCRIPTION**

This command provides the control of calculating power consumption for the domains of interest. Domain-based power consumption data can be generated for a design with multiple power domains. Only the power dissipated in the blocks covered by the current power domain list is calculated and reported.

If you do not use this command, by default, all defined power domains are included in the power analysis.

This command has no effect on power results in non-UPF mode.

For both dynamic (internal and switching) and static (leakage) power, the domain-based power numbers are calculated based on which domain a cell belongs to. Before using this command, user has to define power domains by using the **create\_power\_domain** UPF command. The **get\_current\_power\_domain** command returns the power domains being included in the power analysis.

The following power analysis results are affected by this command in UPF mode:

- Average power report
- Time-based power report
- Power-related attributes:
  - total\_power
  - dynamic\_power

- internal\_power
- switching\_power
- leakage\_power
- intrinsic\_leakage\_power
- gate\_leakage\_power
- x\_transition\_power
- glitch\_power
- peak\_power
- peak\_time

To revert to the default behavior, which includes the whole design covered by all the defined power domains, use the **set\_current\_power\_domain [get\_power\_domain \*]** command.

## EXAMPLES

The following example generates domain-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by the PD1 and PD2 power domains, respectively. The first **report\_power** generates the power analysis results for the whole design, which is the default behavior. The second **report\_power** generates the power analysis results for the power consumed in the PD1 power domain. The third **report\_power** generates the results for the power consumed in the PD2 power domain.

```
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**

```
create_power_domain(2)
get_current_power_domain(2)
get_power_domains(2)
report_power(2)
report_power_domain(2)
```

## **set\_current\_power\_net**

When the variable power\_enable\_multi\_rail\_analysis is true, specifies the power nets defined by UPF create\_supply\_net for power reporting. When false, sets the specific power nets to be included in the power analysis. By default (if not specified), the whole design covered by all the defined supply nets are included in the power analysis and report.

This command has no effect on power results in non-UPF mode.

### **SYNTAX**

```
status set_current_power_net
list_of_power_nets
```

### **Data Types**

*list\_of\_power\_nets*      list

### **ARGUMENTS**

#### *list\_of\_power\_nets*

Specifies the power nets defined in the design to be included in the power analysis. If the *list\_of\_power\_nets* option specifies an undefined supply net, an error is issued, and the current power net setting remains unchanged.

### **DESCRIPTION**

The command behavior depends on the power\_enable\_multi\_rail\_analysis variable. When false, (the default); concurrent multi-rail analysis is disabled; and power for all the power nets in the design are calculated. This command controls which supply nets are to be included in the power analysis. When specified, only the power dissipation on the listed supply nets is calculated and reported. To report power consumption per supply net, the command must be applied for each supply net, and the power reanalyzed per supply net.

When the power\_enable\_multi\_rail\_analysis variable is true, power per supply net is calculated, and the command controls the power reporting and attribute values. When specified, the power dissipation on the listed supply nets is accessible via the get\_attribute and report\_attribute commands, as well as the report\_power command. To access the power attribute values per supply net, the command must be applied for each supply net, before issueing the get\_attribute or report\_attribute commands.

For both dynamic (internal and switching) and static (leakage) power, the power net-based power numbers are calculated based on the PG connection of the cell.

If using the CCS power model in the logic library:

- Dynamic\_currents are based on pg\_pin association.

- Leakage\_currents are based on pg\_pin association.
- Gate leakages are based on the related\_power\_pin for the input related pins.
- Switching powers are based on the related\_power\_pin for the output pins.

If using the NLPM Power model in the technology library with PG pin definitions:

- Internal powers are based on related\_pg\_pin attribute.
- Leakage powers are based on related\_pg\_pin attribute.
- Switching powers are based on the related\_power\_pin for the output pins.

Before using this command, you must define power nets by using the **create\_supply\_net** UPF command. You can use the **report\_supply\_net** command to show the defined power nets. You can use the **report\_pg\_pin\_info** and **report\_power\_network** commands to show the power connections. the **get\_current\_power\_net** command returns the supply nets being included in the power analysis.

The following power analysis results are affected by this command in UPF mode:

- Average power report
- Time-based power report
- Power-related attributes:
  - total\_power
  - dynamic\_power
  - internal\_power
  - switching\_power
  - leakage\_power
  - intrinsic\_leakage\_power
  - gate\_leakage\_power
  - x\_transition\_power

- glitch\_power
- peak\_power
- peak\_time

To revert to the default behavior, which includes the whole design covered by all the defined supply nets, use the **set\_current\_power\_domain [get\_supply\_net \*]** command.

Peak\_power and peak\_time per rail are only accessible when power\_enable\_multi\_rail\_analysis is false. When concurrent multi-rail analysis is performed, only the peak power and peak time for the all the supply nets is accessible. Per rail the peak power attributes are available only when concurrent multi-rail analysis is disabled.

## EXAMPLES

The following example generates power net-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by tje PD1 and PD2 power domains, respectively. There are 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

```
get_current_power_net(2)
set_current_power_domain(2)
get_current_power_domain(2)
create_supply_net(2)
get_supply_nets(2)
report_supply_net(2)
report_pg_pin_info(2)
report_power_network(2)
report_power(2)
```

## **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**

|                     |        |
|---------------------|--------|
| <i>from_object</i>  | object |
| <i>to_object</i>    | object |
| <i>clock_object</i> | object |
| <i>check_value</i>  | 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]
[-models model_list]
```

### **Data Types**

|                   |        |
|-------------------|--------|
| <i>list</i>       | list   |
| <i>name</i>       | string |
| <i>value</i>      | string |
| <i>model_list</i> | 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***  
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**

```
string 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 and 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 or pin.

```
pt_shell> set_disable_clock_gating_check an2/z
pt_shell> set_disable_clock_gating_check or1
```

## **SEE ALSO**

`remove_disable_clock_gating_check(2)`  
`timing_disable_clock_gating_checks(3)`

## **set\_disable\_timing**

Disables timing arcs in a circuit.

### **SYNTAX**

```
string set_disable_timing
[-from from_pin_name -to to_pin_name]
object_list
```

### **Data Types**

|                      |        |
|----------------------|--------|
| <i>from_pin_name</i> | string |
| <i>to_pin_name</i>   | string |
| <i>object_list</i>   | 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 path reported below.

```
pt_shell> report_timing -input_pins

Report : timing
 -path full
 -delay max
 -input_pins
 -max_paths 1
Design : counter
Version: 2002.03-Beta3
Date : Wed Feb 13 12:05:38 2002

```

```
Startpoint: ffb (rising edge-triggered flip-flop)
Endpoint: CO (output port)
Path Group: (none)
Path Type: max
```

| Point                   | Incr | Path   |
|-------------------------|------|--------|
| <hr/>                   |      |        |
| 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   |
| <hr/>                   |      |        |
| (Path is unconstrained) |      |        |

Assuming you want to disable timing arcs through cell **CO**, you can view all timing arcs of **CO** by examining its library cell, **AN2**.

```
pt_shell> report_lib -timing lsi_10k { AN2 }
```

```

Report : library
 -timing_arcs
Library: lsi_10k
Version: 2002.03-Beta3
Date : Thu Feb 14 09:19:43 2002

```

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)

| Lib Cell | Attributes | Arc |                | Arc Pins |    | When |
|----------|------------|-----|----------------|----------|----|------|
|          |            | #   | Type/Sense     | From     | To |      |
| AN2      |            | 0   | positive_unate | A        | Z  |      |
|          |            | 1   | positive_unate | B        | Z  |      |

To disable, the timing arc used in the path reported above, use the **set\_disable\_timing** command as follows:

```
pt_shell> set_disable_timing -from A -to Z [get_cells { CO }]
```

As a result of the above disabling, the **report\_timing** command now finds a new longest path.

```
pt_shell> report_timing
```

```

Report : timing
 -path full
 -delay max
 -max_paths 1
Design : counter
Version: 2002.03-Beta3
Date : Thu Feb 14 09:21:12 2002

```

Startpoint: ffa (rising edge-triggered flip-flop)  
Endpoint: QA (output port)  
Path Group: (none)  
Path Type: max

set\_disable\_timing

1146

| 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

```
remove_disable_timing(2)
report_disable_timing(2)
report_lib(2)
report_timing(2)
reset_design(2)
get_timing_arcs(2)
```

## **set\_distributed\_parameters**

Configures the distributed environment.

### **SYNTAX**

```
status set_distributed_parameters
[-collection_levels collection_level]
[-script script]
```

### **Data Types**

|                         |        |
|-------------------------|--------|
| <i>collection_level</i> | string |
| <i>script</i>           | 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.
```

### **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)
```

## SEE ALSO

```
set_host_options(2)
start_hosts(2)
```

## **set\_distributed\_variables**

Passes Tcl variables from the master to the slaves.

### **SYNTAX**

```
status set_distributed_variables
[-pre_commands pre_command_string]
[-post_commands post_command_string]
object_list
```

### **Data Types**

|                            |        |
|----------------------------|--------|
| <i>pre_command_string</i>  | string |
| <i>post_command_string</i> | string |
| <i>object_list</i>         | list   |

### **ARGUMENTS**

**-pre\_commands *pre\_command\_string***

Specifies a string of commands to be executed in the slave context before the setting of variables. To group commands, use the semicolon (;) character. The maximum size of a command is 1000 chars.

**-post\_commands *post\_command\_string***

Specifies a string of commands to be executed in the slave context after the setting of variables. To group commands, use the semicolon (;) character. The maximum size of a command is 1000 chars.

**object\_list**

Specifies a list of objects.

### **DESCRIPTION**

This command sets variables at the slaves given a set of variables at the master. If the variable to be sent is a simple Tcl list, the Tcl list is recreated at the slaves for all scenarios in focus. If the variable to be sent to the master is an array indexed by the scenario name, a variable is set at the slaves for each scenario in focus that has an entry in the array at the master. In this case, the data can be either a Tcl list or a Tcl array.

### **EXAMPLES**

The following example sets a scenario\_id variable for each scenario and a max\_fanout variable in both scenarios. In the scen1 scenario, the scenario\_id variable is created and set to 1. In the scen2 scenario, the scenario\_id variable is created and set to 2. In both scenarios, the max\_fanout variable is created and set to 10.

```
pt_shell> set max_fanout 10
pt_shell> array set scenario_id {scen1 1 scen2 2}
pt_shell> set_distributed_variables {scenario_id max_fanout}
```

## **SEE ALSO**

`get_distributed_variables(2)`

## **set\_domain\_supply\_net**

Sets the primary power net and primary ground net of an existed power domain.

### **SYNTAX**

```
int set_domain_supply_net
domain_name
-primary_power_net supply_net_name
-primary_ground_net supply_net_name
```

### **Data Types**

|                        |        |
|------------------------|--------|
| <i>domain_name</i>     | string |
| <i>supply_net_name</i> | 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)
help UPF
help "power domains"
```

## **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**

```
string set_dont_touch
object_list [value]
```

### **Data Types**

|                    |         |
|--------------------|---------|
| <i>object_list</i> | list    |
| <i>value</i>       | Boolean |

### **ARGUMENTS**

|                    |                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <i>object_list</i> | Specifies a list of cells, nets, designs, or library cells in which to place the <b>dont_touch</b> attribute.                       |
| <i>value</i>       | Specifies the value in which to set the <b>dont_touch</b> attribute. Allowed values are <i>true</i> (the default) or <i>false</i> . |

### **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** and **create\_timing\_context** commands, and is 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. See the EXAMPLES section for an example.

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

Currently, 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. This limitation will be addressed in an upcoming release.

## 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")
pt_shell>
```

## SEE ALSO

[characterize\\_context\(2\)](#)  
[create\\_timing\\_context\(2\)](#)  
[get\\_attribute\(2\)](#)  
[set\\_dont\\_touch\\_network\(2\)](#)

```
write_context(2)
write_script(2)
```

```
set_dont_touch
1156
```

## **set\_dont\_touch\_network**

Sets the dont\_touch attribute on clock networks for synthesis.

### **SYNTAX**

```
string 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. When a design is optimized, synthesis assigns *dont\_touch* attributes to all cells and nets in the transitive fanout of the *dont\_touch\_network* attributes 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** and **create\_timing\_context** commands and is exported to synthesis using the **write\_context** command.

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* 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]
```

### **SEE ALSO**

[creating\\_timing\\_context\(2\)](#)  
[characterize\\_context\(2\)](#)  
[report\\_context\(2\)](#)  
[write\\_context\(2\)](#)

```
set_dont_touch(2)
```

```
set_dont_touch_network
1158
```

## **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**

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.

**NOTE:** This command must be exercised 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.

**NOTE:** 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_timing(2)`  
`fix_eco_leakage(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**

```
string set_drive
[-rise] [-fall]
[-min] [-max]
resistance_value port_list
```

### **Data Types**

|                         |       |
|-------------------------|-------|
| <i>resistance_value</i> | float |
| <i>port_list</i>        | list  |

### **ARGUMENTS**

*-rise*

Indicates that the *resistance\_value* argument is used to drive the ports only for the rising case.

*-fall*

Indicates 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* variable. The value must be  $\geq 0$ ; units must be the same as those in the technology library.

*port\_list*

Specifies a list of input or inout ports in the current design, in which the *resistance\_value* argument is set.

### **DESCRIPTION**

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

```

| Input | Port | Resistance |      | Transition |      |
|-------|------|------------|------|------------|------|
|       |      | Rise       | Fall | Rise       | Fall |
| <hr/> |      |            |      |            |      |
| 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**

|                   |       |
|-------------------|-------|
| <i>resistance</i> | float |
| <i>port_list</i>  | list  |

### **ARGUMENTS**

|                   |                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------|
| <i>rise</i>       | Specifies to use the resistance to drive the ports for the rising case.                                          |
| <i>-fall</i>      | Specifies to use the resistance to drive the ports for the falling case.                                         |
| <i>-min</i>       | Specifies to use the resistance to drive the ports for the minimum case.                                         |
| <i>-max</i>       | Specifies to use the resistance to drive the ports for the maximum case.                                         |
| <i>resistance</i> | Specifies a nonnegative resistance value for the ports. Port drive resistance (value >= 0).                      |
| <i>port_list</i>  | 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]
[-multiply_by factor]
[-no_design_rule]
[-input_transition_rise rtran]
[-input_transition_fall ftran]
[-clock clock_name]
[-clock_fall]
port_list
```

### **Data Types**

|                      |        |
|----------------------|--------|
| <i>lib_cell_name</i> | string |
| <i>lib_name</i>      | string |
| <i>pin_name</i>      | string |
| <i>form_pin_name</i> | string |
| <i>factor</i>        | float  |
| <i>rtran</i>         | float  |
| <i>ftran</i>         | float  |
| <i>clock_name</i>    | string |
| <i>port_list</i>     | 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.

**-multiply\_by factor**  
Multiplies the calculated transition time by the specified multiplier. The valid transition multiplier range is greater than or equal to 0.

**-no\_design\_rule**  
Specifies not to transfer 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 value 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 value is 0.

**-clock clock\_name**  
The driving cell is set relative the specified clock. This option is deprecated, and will be ignored.

**-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 will be ignored.

**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.

Use the **characterize\_context** command to automatically set driving cell information on subdesign ports based on their context in the entire design. Use the **remove\_driving\_cell** command to remove driving cell information from ports.

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
Version: 1997.08
Date : Tue 1996

```

| Input Port | Rise | Fall | Rise | Fall |
|------------|------|------|------|------|
| A          | --   | --   | --   | --   |

| Input Port | Rise | Fall | Mult | Attrs |
|------------|------|------|------|-------|
| A          | AND  | AND  | --   |       |

## SEE ALSO

[all\\_inputs\(2\)](#)

[set\\_driving\\_cell](#)

1168

```
characterize_context(2)
remove_driving_cell(2)
report_port(2)
reset_design(2)
set_drive(2)
set_load(2)
set_input_transition(2)
```

## **set\_eco\_options**

Specifies options for ECO commands such as **fix\_eco\_timing** and **fix\_eco\_drc**.

### **SYNTAX**

```
status set_eco_options
[-physical_lib_path file_name_list]
[-physical_design_path file_name_list]
[-physical_constraint_file file_name]
[-log_file file_name]
[-mim_group cell_list]
[-filler_cell_names list]
```

### **Data Types**

```
file_name_list list
file_name string
cell_list list
```

### **ARGUMENTS**

**-physical\_lib\_path *file\_name\_list***

Specifies a list of physical library files in Library Exchange Format (LEF). PrimeTime uses the physical library data to understand the physical constraints such as the physical shape of pins, cells, and blocks.

**-physical\_design\_path *file\_name\_list***

Specifies a list of physical data file in Design Exchange Format (DEF). The file must be consistent with the current design. PrimeTime uses the physical design data to understand the design layout details such as available free sites and physical density.

**-physical\_constraint\_file *file\_name***

Specifies a file that contains **create\_voltage\_area** or **create\_placement\_blockage** commands; these commands specify the following physical constraints:

- Voltage areas - In a design with multiple voltage areas, PrimeTime needs to know their locations to place a buffer in the correct voltage area. To create voltage areas, include **create\_voltage\_area** commands in the physical constraint file.

- Placement blockages - If you do not want to insert a buffer or resize a cell in specific areas of your design, create placement blockages by including **create\_placement\_blockage** commands in the physical constraint file.

Note: For best results, include **create\_voltage\_area** and **create\_placement\_blockage** commands in the physical constraint file. Failing to do so might result in poor QoR or incorrect results.

**-log\_file *file\_name***

Specifies a log file that contains warning and error messages generated while

PrimeTime loads the physical library and physical design files. Review the log file if the tool encounters any issues while performing physical ECO fixing.

#### `-mim_group cell_list`

When the **eco\_enable\_mim** variable is true, this option specifies a list of cells that can be treated as multiply instantiated module (MIM) instances. The **write\_changes** command creates one change list per one MIM instance. If only one cell is specified, PrimeTime treats it as one element MIM, and the **write\_changes** command also creates one change list for the instance. Note that this option only specifies the MIM instances. No actual changes are made to the design until you use the **fix\_eco\_timing**, **fix\_eco\_drc**, or **fix\_eco\_power** command. The tool reports any MIM specification errors when it makes changes to the design. You can view the current MIM configuration with the **report\_eco\_options** command. If you are not satisfied with the MIM specification, use the **reset\_eco\_options** command to remove it. Once the **fix\_eco\_timing**, **fix\_eco\_drc**, or **fix\_eco\_power** command applies the changes to the design, the MIM configuration is final, and you cannot change the MIM configuration afterward.

#### `-filler_cell_names list`

Specifies a list of additional filler library cell names that physically-aware ECO commands should treat as filler cells.

## DESCRIPTION

The **set\_eco\_options** command specifies the files that contain physical data to run physical ECO commands such as **fix\_eco\_timing** and **fix\_eco\_drc**. You must use this command before running physical ECO; it enables PrimeTime ADV to understand physical constraints while fixing timing violations.

Like the **link\_path** variable that specifies the files to be linked, the files specified by this command are not loaded into memory at the time of specification. Instead, the **fix\_eco\_timing** or **fix\_eco\_drc** command identifies the files specified by the **set\_eco\_options** command and loads them into memory before fixing timing or design rule constraint (DRC) violations.

### **Distributed Multi-Scenario Analysis**

To execute the **set\_eco\_options** command in all scenarios, use the **remote\_execute** command. Note that you cannot execute the **set\_eco\_options** command in the master process.

### **Specifying MIM groups**

You can break existing MIM group into multiple groups or specify a MIM group for a flat parasitics file. Suppose CPU module is instantiated five times in TOP module as CPU1, CPU2, CPU3, CPU4 and CPU5, and there are CPU.def and CPU.SPEF associated with CPU module. Also you have the following command in your script.

```
read_parasitics -path {CPU1 CPU2 CPU3 CPU4 CPU5} CPU.SPEF
```

When the **eco\_enable\_mim** variable is true, and one of **fix\_eco\_timing**, **fix\_eco\_drc** or **fix\_eco\_power** commands is invoked, PrimeTime detects CPU as MIM. However, you may want to break them into three groups {CPU1 CPU2}, {CPU3 CPU4} and {CPU5} because of large timing differences between the three groups. The following shows how to specify the three groups.

```
set_eco_options -mim_group {CPU1 CPU2}
set_eco_options -mim_group {CPU3 CPU4}
```

Now, suppose you have one flat parasitics in the same design. Thus, PrimeTime cannot detect CPU as MIM. The following illustrates how to group them into a MIM group.

```
set_eco_options -mim_group {CPU1 CPU2 CPU3 CPU4 CPU5}
```

## EXAMPLES

The following example shows how to specify LEF and DEF files. Error and warning messages are recorded in the `error.log` file.

```
pt_shell> set lef_files [glob $my_physical_data/lef/*.lef]
pt_shell> set_eco_options -physical_lib_path $lef_files
 -physical_design_path design.def.gz
 -log_file ./error.log
```

The following example shows how to specify the same information in distributed multi-scenario analysis.

```
remote_execute -verbose {
 set lef_files [glob $my_physical_data/lef/*.lef]
 set_eco_options -physical_lib_path $lef_files
 -physical_design_path design.def.gz
 -log_file ./error.log
}
```

This is an example of a LEF macro filler cell definition.

```
MACRO FILL1TLL
 CLASS CORE SPACER ;
 ORIGIN 0 0 ;
 SIZE 6.4 BY 1.6 ;
```

The keyword **CORE SPACER** in the LEF file is used to denote it is a filler cell. LEF macros of **CLASS CORE** type **FEEDTHRU** are also considered filler cells.

This is an example of a LEF macro cell that is not a filler cell.

```
MACRO FILL4TLL
 ORIGIN 0 0 ;
 SIZE 6.4 BY 1.6 ;
```

This is an example of a DEF instantiation of the FILL4TLL macro.

```
- xofiller!FILL4TLL!11033 FILL43TLL + SOURCE DIST + PLACED (264300 624100) N ;
```

The FILL4TLL macro cell can be treated by physically-aware ECO commands as filler cells by providing it to the `filler_cell_names` option as under:-

```
set user_filler_cells "FILL4TLL"
```

```
set_eco_options
```

```
set_eco_options -physical_lib_path $lef_files
 -physical_design_path design.def.gz
 -filler_cell_names $user_filler_cells
 -log_file ./error.log
```

When the eco\_allowFillerCellsAsOpenSites variable is set to true, physically-aware ECO commands treat these user defined filler cells in addition to other automatically detected LEF filler cells as open\_sites.

Suppose CPU module is instantiated four times: CPU1, CPU2, CPU3 and CPU4. The following example shows how to specify MIM for the CPU module.

```
pt_shell> set_eco_options -mim_group {CPU1 CPU2 CPU3 CPU4}
```

## SEE ALSO

```
create_placement_blockage(2)
create_voltage_area(2)
fix_eco_drc(2)
fix_eco_timing(2)
report_eco_options(2)
reset_eco_options(2)
write_changes(2)
eco_enable_mim(3)
```

## **set\_equal**

Sets two ports to be logically equivalent.

### **SYNTAX**

```
string set_equal
port1
port2
```

### **Data Types**

```
port1 string
port2 string
```

### **ARGUMENTS**

```
port1
 Specifies the first input port.

port2
 Specifies another 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.

Use the **reset\_design** command to remove this property from a port.

This attribute is characterized by the **characterize\_context** and **create\_timing\_context** commands and is 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

[set\\_opposite\(2\)](#)  
[creating\\_timing\\_context\(2\)](#)  
[characterize\\_context\(2\)](#)  
[write\\_context\(2\)](#)

## **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**

```
Boolean 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**

|                          |        |
|--------------------------|--------|
| <i>from_list</i>         | list   |
| <i>rise_from_list</i>    | list   |
| <i>fall_from_list</i>    | list   |
| <i>through_list</i>      | list   |
| <i>rise_through_list</i> | list   |
| <i>fall_through_list</i> | list   |
| <i>to_list</i>           | list   |
| <i>rise_to_list</i>      | list   |
| <i>fall_to_list</i>      | list   |
| <i>comment_string</i>    | 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.

**-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.

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. When the clocks are asynchronous to each other, the **set\_clock\_groups -async** command should be used instead of the **set\_false\_paths** command between the async clocks. Without the **set\_clock\_groups -async** command, the clocks are treated to be sync 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)
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\_fanout\_load**

1180

## **set\_host\_options**

Specifies host options for compute resources.

### **SYNTAX - Current (master) process**

```
status set_host_options
[-local_process]
[-max_cores number]
[-protocol auto | lsf | sh | rsh | ssh | sge | rtida | pbs]
```

### **SYNTAX - Remote (slave) processes**

```
status set_host_options
[-max_cores number]
[-name name]
[-submit_command command]
[-terminate_command command]
[-num_processes number]
[host_names]
```

### **Data Types**

|                   |         |
|-------------------|---------|
| <i>number</i>     | integer |
| <i>name</i>       | string  |
| <i>command</i>    | string  |
| <i>host_names</i> | list    |

## **ARGUMENTS**

### **-local\_process**

Use this option to override the default resource usage limits for the current process. If neither the **-local\_process** nor **-num\_processes** option are specified, the **-local\_process** option is implied default.

### **-max\_cores *number***

Use this option to modify the CPU cores usage limit for the current run if the **-local\_process** option is specified or the subsequently launched remote process otherwise. The *number* value is currently limited to the range between 1 and 16.

### **-protocol auto | lsf | sh | rsh | ssh | sge | rtida | pbs**

Specifies the type of the communication protocol. The default is **auto**. If you do not specify this option, the communication protocol is inferred from the **-submit\_command** *command* option.

### **-name *name***

Use this option to specify the unique name for the host options being defined. Any valid user-defined string can be specified; however, if options of the same name already exist, the previously defined options are removed and

replaced with the new options. Any remote processes started using the previous options are shutdown. This option cannot be specified in conjunction with the **-local\_process** option.

**-submit\_command command**

Use this option to specify the command to call when submitting and launching the PrimeTime processes associated with the host options. This option specifies the path and command to call to perform this operation. When it is called, the job/process ID of the job is captured and associated with the host options. The job and process ID can be reported using the **report\_host\_usage** command. If the **-submit\_command** and *host\_name* options are not specified or the *host\_name* is the same as where the master process is running or is localhost, the processes are launched locally by the operating system of the master. However, if the *host\_name* is specified and does not meet the preceding criterion, rsh is used to make remote connections to the host when setting up the processes. If a command is specified, it should be noninteractive in nature. This option cannot be specified in conjunction with the **-local\_process** option.

**-terminate\_command command**

Use this option to specify the command to call to terminate the jobs submitted by calling the submit command. When it is necessary to terminate the jobs (for example, at exit) if the PrimeTime processes have not come online, the master uses the terminate command to terminate the jobs by calling the command and passing the job ID captured from the submit command. This option cannot be specified in conjunction with the **-local\_process** option.

**-num\_processes number**

Use this option to specify the number of processes to launch. This option cannot be specified in conjunction with the **-local\_process** option.

**host\_names**

Use this option to specify a list of names of hosts on which to launch the slave processes. If nothing is specified, by default a list containing a single entry called localhost is assumed. The default name for the host on which the current process is running is called localhost. If host names are specified and if any of the names are not localhost or the name of the host on which the master process is running, the **-submit\_command** option can be used to specify the command to setup a remote connection to the hosts in order to launch the PrimeTime slave processes. This option cannot be specified in conjunction with the **-local\_process** option.

## DESCRIPTION

This command is used to specify the host options of the current process in conjunction with the **-local\_process** option.

This command is also used in conjunction with the **-num\_processes** option to set up the host options to use when launching PrimeTime processes. The command specifies the options but does not launch the processes. For more information about launching the processes, see the **start\_hosts** command.

If neither the **-num\_processes** nor **-local\_process** option is specified then **-local\_process** is the implied default.

The host cores usage limit can be modified using the **set\_host\_options** command. However, the initial limits are set as follows. For the current or local process, the cores usage limit is dictated by the minimum of the limit afforded by a single PrimeTime license usage as well as the physical number of cores. The default single license is equivalent to a cores limit of 4. So, if you launch a PrimeTime shell on a 2-core machine, the initial cores limit is 2. Whereas the cores usage limit would be 4 if the machine had 8 cores. For a slave process in a distributed multicore analysis run, the initial cores usage limit is 1. While a slave process in a distributed multi-scenario run has an initial usage limit of 4.

To use a custom wrapper for launching slave processes, see the **set\_distributed\_parameters** command.

## EXAMPLES

In the following example, the master process is running on the host named ptopt018 using a 64-bit binary and specifies five different sets of host options:

- my\_opts1 - Specifies one process with the same architecture as the master on the same host as the master and does not specify an upper limit on the number of cores to use; therefore, the default applies.
- my\_opts2 - Specifies two processes on the same host as the master but explicitly mentions the name of the master's host and specifies to use a maximum of 2 cores per process.
- my\_opts3 - Specifies four processes (with the same architecture as the master) on the host named ptopt010 using "rsh" to connect to ptopt010 and specifying to use a maximum of 3 cores per process.
- my\_opts4 - Specifies five processes (with the same architecture as the master) on an LSF farm, requesting 500 MB of memory and 2 slots per compute server and specifying to use a maximum of 2 cores per process. Note that the farm job must be specified with the number of slots needed to support the number of cores to use, therefore the "-n 2 -R span\[ptile=2\]". A command is also provided for the termination of jobs that do not come online.
- my\_opts5 - Specifies two processes (with the same architecture as the master) on a Grid farm requiring the jobs to be launched using the project named "bnormal" and does not specify an upper limit on the number of cores to use, therefore the default applies. A command is also provided for the termination of jobs that do not come online. Note: The **-b y** option is required when accessing grid.

```
pt_shell> set_host_options -name my_opts1
1
pt_shell> set_host_options -name my_opts2 -num_processes 2 \
 ptopt018
1
pt_shell> set_host_options -name my_opts3 -num_processes 4 \
```

```

 -submit_command "/usr/bin/rsh -n" ptopt010
1
pt_shell> set_host_options -name my_opts4 -num_processes 5 \
 -submit_command "/lsf/bin/bsub -n 2 -R rusage\[mem=500\] -
R span\[ptile=2\]" \
 -terminate_command "/lsf/bin/bkill"
1
pt_shell> set_host_options -name my_opts5 -num_processes 2 \
 -submit_command "/grid/bin/qsub -b y -P bnormal" \
 -terminate_command "/grid/bin/qdel"
1
pt_shell> report_host_usage -verbose

Report : host_usage
 -verbose
...

```

| Options  | Host          | 32Bit | Num       |
|----------|---------------|-------|-----------|
| Name     |               |       | Processes |
| my_opts1 | **localhost** | N     | 1         |
| my_opts2 | **ptopt018**  | N     | 2         |
| my_opts3 | ptopt010      | N     | 4         |
| my_opts4 | >>farm<<      | N     | 5         |
| my_opts5 | >>farm<<      | N     | 2         |

| Options  | Host          | Action    | Command                                                     |
|----------|---------------|-----------|-------------------------------------------------------------|
| Name     | Name          |           |                                                             |
| my_opts1 | **localhost** | SUBMIT    | <execute directly on masters host>                          |
| my_opts2 | **ptopt018**  | SUBMIT    | <execute directly on masters host>                          |
| my_opts3 | ptopt010      | SUBMIT    | /usr/bin/rsh -n                                             |
| my_opts4 | >>farm<<      | SUBMIT    | /lsf/bin/bsub -n 2 -R rusage\[mem=500\] - R span\[ptile=2\] |
|          |               | TERMINATE | /lsf/bin/bkill                                              |
| my_opts5 | >>farm<<      | SUBMIT    | /grid/bin/qsub -b y -P bnormal                              |
|          |               | TERMINATE | /grid/bin/qdel                                              |

#### Hosts limits:

| Name                | User | Cores limits |           |
|---------------------|------|--------------|-----------|
|                     |      | Hardware     | Effective |
| ** local process ** | none | 4            | 4         |
| my_opts1            | 1    | -            | -         |
| my_opts2            | 2    | -            | -         |
| my_opts3            | 3    | -            | -         |
| my_opts4            | 2    | -            | -         |
| my_opts5            | 1    | -            | -         |

1

In the following example, PrimeTime is launched onto a 16-core machine. Initially, reporting usage shows that the user local process is effectively limited to 4 cores

even though no user limit is set. Subsequently, you can up the cores limit to 8 by issuing the appropriate **set\_host\_options** command. Note that this causes additional licenses to be checked out as indicated by the informational messages issued.

```
pt_shell> report_host_usage -verbose
```

```

Report : host_usage
-verbose
...

```

Hosts limits:

| Name                | Cores limits |          |           |
|---------------------|--------------|----------|-----------|
|                     | User         | Hardware | Effective |
| ** local process ** | none         | 16       | 4         |

1

```
pt_shell> set_host_options -max_cores 8
```

```
Information: Checked out license 'PrimeTime' (PT-019)
Information: The specified max_cores value increases the per-feature
license count by 1 licenses. (CMCR-103)
```

1

```
pt_shell> report_host_usage -verbose
```

```

Report : host_usage
-verbose
...

```

Hosts limits:

| Name                | Cores limits |          |           |
|---------------------|--------------|----------|-----------|
|                     | User         | Hardware | Effective |
| ** local process ** | 8            | 16       | 8         |

1

## SEE ALSO

```
remove_host_options(2)
report_host_usage(2)
set_distributed_parameters(2)
start_hosts(2)
stop_hosts(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**

```
int 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**

|                    |       |
|--------------------|-------|
| <i>value</i>       | float |
| <i>object_list</i> | 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**

|                      |       |
|----------------------|-------|
| <i>clock_name</i>    | list  |
| <i>pin_port_name</i> | list  |
| <i>delay_value</i>   | float |
| <i>port_pin_list</i> | 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 respect 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**

|                     |       |
|---------------------|-------|
| <i>width_value</i>  | float |
| <i>height_value</i> | float |
| <i>object_list</i>  | 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**

|                   |        |
|-------------------|--------|
| <i>clock_name</i> | string |
| <i>transition</i> | float  |
| <i>port_list</i>  | 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\_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]
 [-force_isolation]
 [-name_prefix prefix_string]
 [-name_suffix suffix_string]
 isolation_strategy
```

### **Data Types**

|                               |        |
|-------------------------------|--------|
| <i>power_domain</i>           | string |
| <i>supply_net_name</i>        | string |
| <i>supply_set_name</i>        | string |
| <i>objects</i>                | list   |
| <i>source_supply_set_name</i> | string |
| <i>sink_supply_set_name</i>   | string |
| <i>prefix_string</i>          | string |
| <i>suffix_string</i>          | string |
| <i>isolation_strategy</i>     | 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.

**-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 shutdown_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 location]
[-isolation_sense sense]
isolation_name
```

### **Data Types**

|                       |        |
|-----------------------|--------|
| <i>domain</i>         | string |
| <i>signal</i>         | string |
| <i>location</i>       | string |
| <i>sense</i>          | string |
| <i>isolation_name</i> | 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.

-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*.

-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\_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**

```
int set_level_shifter_strategy
 -rule strategy
```

### **ARGUMENTS**

**-rule *strategy***

Specifies types of source-sink pairs of mismatching signal levels that the **check\_timing -include signal\_level** command reports.

The allowed values are *all*, *low\_to\_high*, and *high\_to\_low*. The *all* value is the default strategy.

Using the **-rule low\_to\_high** option reports the voltage level mismatches when a source at a lower voltage drives a sink at a higher voltage.

Using the **-rule high\_to\_low** option 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**

```
int set_level_shifter_threshold
-voltage volt
-percent diff
```

### **Data Types for All Modes**

|             |       |
|-------------|-------|
| <i>volt</i> | float |
| <i>diff</i> | 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**

|                          |      |
|--------------------------|------|
| <i>lib_cells</i>         | list |
| <i>library_pin_names</i> | list |
| <i>power_rail_names</i>  | list |

### **ARGUMENTS**

```
-lib_cell_names lib_cells
 Specifies a list of library cells on power_rails are to be recognized.

-lib_pin_names library_pin_names
 Specifies the names of library pin to be annotated.

-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 lib cell. When linking physical Verilog to a logical library, pins specified by this command can be present in the Verilog, but not in the logical library.

### **EXAMPLES**

The following example shows how to set a physical power pin on a library cell to link with to logical 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 (logical) library *typ*. This allows successful linking of physical Verilog to logical libraries in which power pins not specified.

## **set\_library\_driver\_waveform**

Sets the driver waveform used to characterize the timing library.

### **SYNTAX**

```
int set_library_driver_waveform
-type type
[lib_objs]
```

### **Data Types**

*lib\_objs*            list

### **ARGUMENTS**

**-type type**  
The type of the waveforms. It has two possible values, *ramp*, which is linear waveforms and *standard*, which is 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)`  
`si_ccs_use_gate_level_simulation(3)`

## **set\_load**

Sets the capacitance to a specified value on the specified ports and nets in the current design.

### **SYNTAX**

```
Boolean set_load
[-min]
[-max]
[-rise]
[-fall]
[-pin_load]
[-wire_load]
[-subtract_pin_load]
capacitance
objects
```

### **Data Types**

|                    |       |
|--------------------|-------|
| <i>capacitance</i> | float |
| <i>objects</i>     | 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. With this option, the total capacitance computed on these nets during the **update\_timing** command does not include pin capacitances. Use this option only if *capacitance* includes pin capacitances.

#### **capacitance**

Specifies the capacitance value to be set on the ports and nets in *objects*. It is in the units of the technology library.

#### **objects**

Specifies a list of ports and nets in the current design, on which the *capacitance* is to be set. Note that 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 issued 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 port named *the\_answer*.

```
pt_shell> set_load 2 the_answer
```

The following example sets a capacitance of 2.5 units (the estimated wire capacitance) plus the capacitance of three inverter input pins from the library named *tech\_lib* 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 pin *Z* 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 more, 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 port named *the\_answer*.

```
pt_shell> set_load -wire_load 5 the_answer
```

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(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 ( $\geq 0$ ). 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\_area**

1220

## **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_max_capacitance`

1222

```
set_min_capacitance(2)
```

set\_max\_capacitance  
1223

## **set\_max\_delay**

Specifies a maximum delay for timing paths.

### **SYNTAX**

```
Boolean set_max_delay
[-rise]
[-fall]
[-reset_path]
[-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**

|                          |        |
|--------------------------|--------|
| <i>from_list</i>         | list   |
| <i>rise_from_list</i>    | list   |
| <i>fall_from_list</i>    | list   |
| <i>to_list</i>           | list   |
| <i>rise_to_list</i>      | list   |
| <i>fall_to_list</i>      | list   |
| <i>through_list</i>      | list   |
| <i>rise_through_list</i> | list   |
| <i>fall_through_list</i> | list   |
| <i>comment_string</i>    | string |
| <i>delay_value</i>       | 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.

**-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.

**-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.

**-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.

**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 technology 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 end points 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 clock -to pin`
3. `set_max_delay -from pin -to clock`
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**

|                     |       |
|---------------------|-------|
| <i>fanout_value</i> | float |
| <i>object_list</i>  | list  |

### **ARGUMENTS**

|                     |                                                                                          |
|---------------------|------------------------------------------------------------------------------------------|
| <i>fanout_value</i> | Fanout limit (Value $\geq 0$ ). This is the maximum fanout load in library fanout units. |
| <i>object_list</i>  | 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_fanout`  
1230

## **set\_max\_time\_borrow**

Limits time borrowing for latches.

### **SYNTAX**

```
string 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. Defines the desired 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  $(\text{closing\_edge} - \text{open\_edge})$ . 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 technology 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\(2\)](#)  
[remove\\_max\\_time\\_borrow\(2\)](#)  
[get\\_attribute\(2\)](#)  
[report\\_exceptions\(2\)](#)

[set\\_max\\_time\\_borrow](#)  
1232

## **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**

|                         |       |
|-------------------------|-------|
| <i>transition_value</i> | float |
| <i>object_list</i>      | 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  $\geq 0$ ). 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)
```

**set\_max\_transition**

1234

```
report_port(2)
```

```
set_max_transition
1235
```

## **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 an 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)
```

**set\_message\_info**

1236

...  
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**

|                          |       |
|--------------------------|-------|
| <i>capacitance_value</i> | float |
| <i>object_list</i>       | list  |

### **ARGUMENTS**

*capacitance\_value*

Specifies the capacitance limit (Value  $\geq 0$ ). 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**

```
Boolean set_min_delay
[-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]
delay_value
```

### **Data Types**

|                          |        |
|--------------------------|--------|
| <i>from_list</i>         | list   |
| <i>rise_from_list</i>    | list   |
| <i>fall_from_list</i>    | list   |
| <i>through_list</i>      | list   |
| <i>rise_through_list</i> | list   |
| <i>fall_through_list</i> | list   |
| <i>to_list</i>           | list   |
| <i>rise_to_list</i>      | list   |
| <i>fall_to_list</i>      | list   |
| <i>comment_string</i>    | string |
| <i>delay_value</i>       | 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.

**-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.

**-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.

**-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.

**-rise\_through** *rise\_through\_list*

This option is similiar 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 similiar 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.

**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 technology 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 end points 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 clock -to pin`
3. `set_min_delay -from pin -to clock`
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**

|                    |        |
|--------------------|--------|
| <i>min_library</i> | string |
| <i>max_library</i> | 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)
```

**set\_min\_pulse\_width**

1248

```
report_pulse_clock_min_width(2)
reset_design(2)
set_pulse_clock_min_width(2)
timing_enable_pulse_clock_constraints(3)
```

set\_min\_pulse\_width  
1249

## **set\_mode**

Selects the active mode of cell mode groups or design mode groups

### **SYNTAX**

```
Boolean set_mode
[-type cell | design]
[mode_list]
[instance_list]
```

### **Data Types**

|                      |      |
|----------------------|------|
| <i>mode_list</i>     | list |
| <i>instance_list</i> | 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_mode
1252
```

## **set\_multi\_input\_switching\_coefficient**

Sets delay and slew coefficients for multi-input switching (MIS) analysis for a specified list of library cells.

### **SYNTAX**

```
int set_multi_input_switching_coefficient
[-rise]
[-fall]
-delay coeff_value
[-slew coeff_value]
object_list
```

### **Data Types**

|             |       |
|-------------|-------|
| coeff_value | float |
| object_list | list  |

### **ARGUMENTS**

-rise  
Applies the coefficients to rise transitions only.

-fall  
Applies the coefficients to fall transitions only.

-delay coeff\_value  
Indicates the base coefficient of the cell delay for multi-input switching analysis when the specified library cell is used.

-slew coeff\_value  
Indicates the base coefficient of the delta delay for multi-input switching analysis when the specified library cell is driving the victim net. Note that this option does not affect the cell slew.  
The slew base coefficient is optional. The default is 1.

object\_list  
Specifies a list of library cells for multi-input switching analysis.

### **DESCRIPTION**

This command sets the base coefficients for the cell delay and delta delay of a victim net driven by this cell during multi-input switching (MIS) analysis. To enable multi-input switching analysis, set the **si\_enable\_multi\_input\_switching\_analysis** variable to true; the default of this variable is false.

If the **si\_enable\_multi\_input\_switching\_timing\_window\_filter** variable is set to true, the tool uses the base coefficient value to derive the actual MIS factor by considering the arrival windows of the input pins and their overlap analysis.

If the `si_enable_multi_input_switching_timing_window_filter` variable is set to false, the base coefficient value is directly applied to the MIS factor.

The tool uses the slew base coefficient to derive the delta delay MIS factor for a victim net driven by the specified library cell. Note that the slew MIS factor is not subject to multi-input switching analysis. Therefore, the output slew does not change. The tool derives and uses the actual slew MIS factor to calibrate the delta delay in the following ways:

- Aggressor bump size
- Victim driving strength

If you use the `set_multi_input_switching_coefficient` command multiple times on the same library cell, the last specified coefficient overwrites the previously specified coefficients.

## EXAMPLES

The following example sets the delay coefficient value of 0.7 for all instances of library cell AND2 in the library MY\_LIB:

```
pt_shell> set_multi_input_switching_coefficient -delay 0.7 [get_lib_cells MY_LIB/AND2]
```

## SEE ALSO

```
report_multi_input_switching_coefficient(2)
reset_multi_input_switching_coefficient(2)
si_enable_multi_input_switching_analysis(3)
si_enable_multi_input_switching_timing_window_filter(3)
```

## **set\_multi\_scenario\_license\_limit**

Specifies the upper limit on the number of licenses, of a particular feature, the master in multi-scenario analysis can checkout for dynamic allocation among its slave processes.

### **SYNTAX**

```
int set_multi_scenario_license_limit
 -feature feature_name
 [-force]
 number
```

### **Data Types**

|                     |        |
|---------------------|--------|
| <i>feature_name</i> | string |
| <i>number</i>       | int    |

### **ARGUMENTS**

-feature *feature\_name*  
Select the feature to be dynamically managed for multi-scenario analysis. See the help -v output of the command for the currently supported features.

*number*  
Specifies the maximum number of licenses to be checked out for slave usage. To remove license limit, set the number to -1.

-force  
Forces licenses to be checked out straight away. If this option is not specified licenses are only checked out when needed.

### **DESCRIPTION**

This command is only available when you invoke PrimeTime with the *-multi\_scenario* option. For example, **pt\_shell -multi\_scenario**.

In multi-scenario analysis, it is possible to specify, using the **set\_multi\_scenario\_license\_limit** command, the maximum number of feature licenses for slave usage during the analysis. Licenses can be checked out up front using the -force option. Otherwise, by default, licenses are only checked out on an as-needed basis. After the master checks out the licenses, it does not release them until it exits. The master dynamically allocates the licenses amongst the slave processes on an as-needed basis. Therefore, more slave processes can be started than there are licenses checked out. However, at any point in time, the number of executing slave processes never exceeds the number of licenses checked out by the master for slave usage.

If the -force option is issued the master tries to immediately checkout up to the limit specified for the particular feature. It might checkout fewer licenses, but it never attempts to checkout more.

If licenses are being checked out on an as-needed basis and a request for a license checkout succeeds, a message is printed to the master shell and the analysis continues.

When the master checks out N licenses of a feature this implies that N slave processes can operate concurrently as the master does not require a license itself during the analysis.

## EXAMPLES

In the following example, the upper limit on the number of PrimeTime licenses is set to seven and the number of PrimeTime SI licenses set to four. No licenses are checked out at the point at which the commands are called, the licenses are checked out as and when the slaves need them during the subsequent analysis.

```
pt_shell> set_multi_scenario_license_limit -feature PrimeTime 7
Information: Setting license limit for feature 'PrimeTime' to 7. (PT-021)
1
pt_shell> set_multi_scenario_license_limit -feature PrimeTime-SI 4
Information: Setting license limit for feature 'PrimeTime-SI' to 4. (PT-021)
1
```

## SEE ALSO

`report_multi_scenario_design(2)`

## **set\_multicycle\_path**

Defines the multicycle path.

### **SYNTAX**

```
Boolean 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.

**-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.

**-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.

**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 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 multi-frequency 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 phil to phil to be zero cycles.

```
pt_shell> set_multicycle_path 0 -from phil -to phil
```

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\_noise\_derate**

Sets noise derating information for the current design.

### **SYNTAX**

```
int set_noise_derate
[-above]
[-below]
[-low]
[-high]
[-height_offset hoffset]
[-height_factor hfactor]
[-width_factor wfactor]
[object_list]
```

### **Data Types**

|                    |       |
|--------------------|-------|
| <i>hoffset</i>     | float |
| <i>hfactor</i>     | float |
| <i>wfactor</i>     | float |
| <i>object_list</i> | 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.

-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. 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"**

|                     |       |
|---------------------|-------|
| <i>height_value</i> | float |
| <i>width_value</i>  | float |
| <i>area_value</i>   | float |
| <i>object_list</i>  | 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,  $\text{bump\_height} > C1 + C2 / (\text{bump\_width} - C3)$

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  $\text{bump\_height} > 0.59 + 0.0064 / (\text{bump\_width} - 0.04)$ .

## 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**

|                     |       |
|---------------------|-------|
| <i>margin_value</i> | float |
| <i>object_list</i>  | 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_noiseViolationSources(2)
reportNoise(2)
reportNoiseParameters(2)
reportNoiseViolationSources(2)
resetNoiseParameters(2)
updateNoise(2)
```

## **set\_ocvm\_table\_group**

Associates an AOCV or POCV named table group or Liberty-based AOCV or POCV derating group with the list of specified design objects.

### **SYNTAX**

```
status set_ocvm_table_group
ocvm_table_group_name
-type aocvm | pocvm
[-table_type coefficient | distance]
[object_list]
```

### **Data Types**

|                              |        |
|------------------------------|--------|
| <i>ocvm_table_group_name</i> | string |
| <i>object_list</i>           | list   |

### **ARGUMENTS**

*ocvm\_table\_group\_name*  
Specifies the name of the AOCV or POCV table group.

*-type aocvm | pocvm*  
Specifies AOCV or POCV tables.

*-table\_type coefficient | distance*  
Specifies coefficient or distance-based POCV tables. The default is **coefficient**. Do not use this option for AOCV.

*object\_list*  
Specifies the list of design objects.

### **DESCRIPTION**

The **set\_ocvm\_table\_group** command can operate on either hierarchical cells / top level design or on lib\_cells.

When operating on hierarchical cells or a top-level design, the command sets the association between a named AOCV or POCV table group and a hierarchical instance. An AOCV or POCV table group is a collection of AOCV or POCV tables with the same name. Names are specified using the group\_name field in the AOCV or POCV table syntax. When associations are specified between hierarchical cells in the design and named AOCV or POCV table groups, a cell (or net) derives its AOCV or POCV derating from the table group associated with the hierarchical cell (or design) that (a) fully contains the cell (or net), and (b) is the lowest-level (closest ancestor) hierarchical cell (or design) with an associated AOCV or POCV table group. If no such hierarchical cell (or design) containing the cell (or net) is found, the unnamed AOCV or POCV table group is used. Note that a net inherits the AOCV or POCV derating set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV or POCV table group), the lowest-level hierarchical cell that fully

contains the net is not the block. Therefore, the net derating used comes from an AOCV or POCV table group that contains the block and fully encloses the net. Note that for POCV, table groups for coefficients and distance-based tables are independent.

When operating on lib\_cells the **set\_ocvm\_table\_group** command can be used to assign a Liberty based AOCVM table from a Liberty library to the given lib cell. The Liberty AOCVM format allows for tables, which are not assigned to any library cell by default, to still be referenced by name. This command provides the ability to setup this reference. The syntax for the P/AOCVM derate group to be referenced follows this pattern:

```
lib_name/[lib_cell_name]/derate_group_name
```

The *lib\_cell\_name* is optional and applicable only for lib\_cell based table. A lib\_cell based can only be assigned to a library cell of the same name.

## EXAMPLES

In the following example, the **set\_ocvm\_table\_group** command performs an association between the hierarchical cell H1 and the named POCV coefficient table group A1. A POCV coefficient table belonging to the A1 table group is also shown.

```
pt_shell> set_ocvm_table_group -type pocvm -table_type coefficient A1 [get_cells H1]
pt_shell> sh cat test.pocvm

version: 4.0
ocvm_type: pocvm
group_name: A1
object_type: lib_cell
rf_type: rise
delay_type: cell
derate_type: late
object_spec: my_lib/*
coefficient: 0.05
```

## SEE ALSO

```
get_timing_paths(2)
report_ocvm(2)
report_timing(2)
reset_ocvm_table_group(2)
```

## **set\_operating\_conditions**

Defines the operating conditions or environmental characteristics for the current design.

### **SYNTAX**

```
int 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**

|                      |        |
|----------------------|--------|
| <i>lib</i>           | string |
| <i>condition</i>     | string |
| <i>min_condition</i> | string |
| <i>max_condition</i> | string |
| <i>min_lib</i>       | string |
| <i>max_lib</i>       | string |
| <i>objects</i>       | list   |

### **ARGUMENTS**

**-analysis\_type single | on\_chip\_variation**

Indicates how the specified operating conditions are to be used. This option has the following mutually-exclusive arguments:

- **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
 Specifies the cells or ports on which to set operating conditions. If you do
 not use this option, operating conditions are set on the 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

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 operating conditions set on the cells or blocks library, 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, 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 BCCOM values are used for minimum operating conditions and the WCCOM values are used for maximum operating conditions. When reporting maximum delays, the delays are computed for the maximum operating conditions. When reporting minimum delays, the delays are computed for the minimum operating

condition.

```
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**

```
string set_opposite
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 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** and **create\_timing\_context** commands and is 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\(2\)](#)  
[creating\\_timing\\_context\(2\)](#)  
[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**

|                      |        |
|----------------------|--------|
| <i>clock_name</i>    | list   |
| <i>pin_port_name</i> | list   |
| <i>group_name</i>    | string |
| <i>delay_value</i>   | float  |
| <i>port_pin_list</i> | 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**

```
Boolean 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\_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 supply_set_ref]
[-repeater_supply supply_set_ref]
[-attribute name value]*
```

### **Data Types**

|                       |        |
|-----------------------|--------|
| <i>port_list</i>      | list   |
| <i>element_list</i>   | list   |
| <i>supply_set_ref</i> | string |
| <i>name</i>           | string |
| <i>value</i>          | 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*

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 unsupported 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 'snsr\_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**

|                      |      |
|----------------------|------|
| <i>fanout_number</i> | int  |
| <i>port_list</i>     | 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\_analysis\_options**

Sets the options for power analysis.

### **SYNTAX**

```
int set_power_analysis_options
[-static_leakage_only]
[-variation_quantile quantile]
[-waveform_interval sampling_interval]
[-cycle_accurate_cycle_count cycles]
[-cycle_accurate_clock clock]
[-waveform_format fsdb | out | none]
[-waveform_output file_prefix]
[-include top | all_without_leaf | all_with_leaf]
[-include_groups group_list]
[-cells cell_list]
[-sdpd_tracking enabled | disabled]
[-sdpd_tracking_cells cell_list]

float quantile
float sampling_interval
int cycles
string clock
string file_prefix
list group_list
list cell_list
```

### **ARGUMENTS**

**-static\_leakage\_only**  
Specifies that averaged power analysis will only calculate static leakage power and skip dynamic power calculation. The option only applies to averaged power analysis.

**-variation\_quantile *quantile***  
Specifies that the leakage variation report should report the specified leakage quantile in the quantile column of the report. The value is between 0.0 and 1.0. Default value is 0.99. This option only applies to statistical leakage power variation analysis.

**-waveform\_interval *sampling\_interval***  
Specifies the sampling interval that is used for power waveform. The default value is the timescale from VCD file. This option only applies to time\_based power analysis. The option will be ignored if VCD comes from RTL or zero delay simulation.

**-cycle\_accurate\_cycle\_count *cycles***  
Specifies the number of clock cycle over which the power consumed by the design is considered constant when VCD is RTL or zero delay. Default value is 1. This option only applies to time\_based power analysis, specifically for RTL or zero delay VCD or its equivalents (VPD etc.).

**-cycle\_accurate\_clock *clock***  
 Specifies the reference clock for time\_based power analysis, when VCD is RTL or zero delay. The power consumed by the design is considered constant over one or more cycles of the specified clock. The default is the fastest clock in the design. This option only applies to time\_based power analysis, specifically for RTL or zero delay VCD or its equivalents (VPD etc.).

**-waveform\_format *fsdb / out / none***  
 Specify the format of the file in which the waveform data is to be written. The value for this option can be: *fsdb*, *out* or *none*. Default is *fsdb*. If it's set to *none*, waveform data is not to be dumped, but peak power is still calculated. This option only applies to time\_based power analysis.

**-waveform\_output *file\_prefix***  
 Specify the prefix of the file in which the waveform data is to be written. The default is *primetime\_px*. This option only applies to time\_based power analysis.

**-include *top / all\_without\_leaf / all\_with\_leaf***  
 Specifies the objects in the design hierarchy to be monitored for waveform generation. The tool will only be able to show the power waveforms limited by this option. There are three allowed values for the option. If the value is *top*, only top level design is monitored for power waveform generation. If the value is *all\_without\_leaf*, all design hierarchies except leaf cells are monitored. If the value is *all\_with\_leaf*, all design hierarchies including leaf cells are monitored. The default value is *all\_without\_leaf*. This option only applies to time\_based power analysis.

**-include\_groups *group\_list***  
 Specifies the power groups to be monitored for power waveform generation. The tool will be able to show the power waveforms for the specified power groups. By default, no power group is monitored. This option only applies to time\_based power analysis.

**-cells *cell\_list***  
 Specifies the cell names or collections of cells for which power needs to be calculated. By default, PrimeTime PX calculates power for the whole design.

**-sdpd\_tracking *enabled / disabled***  
 Enables or disables SDPD activity tracking for cells specified in the -  
*sdpd\_tracking\_cells* list. The default is disabled.

**-sdpd\_tracking\_cells *cell\_list***  
 Specifies the list of cells for which SDPD activity is to be tracked. The default list contains all black box and memory cells.

## DESCRIPTION

There are several power analysis mode, such as *averaged*, *time\_based*, *leakage\_variation*, in PrimeTime PX. The mode is specified by the **power\_analysis\_mode** variable before the major power analysis commands. The working command for all power analysis mode is **update\_power**. **update\_power** itself does not take any option. The **set\_power\_analysis\_options** command is used between **power\_analysis\_mode** variable and **update\_power** command to specify various options for different modes of power

analysis in PrimeTime PX. The options from this command will control how power analysis of the specified mode behaves.

Each option may be used for one or more modes of power analysis. If an option is not supposed to be used in a power analysis mode, the command will stop and an error message will be issued. **set\_power\_analysis\_options** with no option will reset all the power analysis options to default values. New issue of **set\_power\_analysis\_options** will overwrite the previous one and reset the rest of power analysis options to default values. Option **-cycle\_accurate\_clock** and **-cycle\_accurate\_cycle\_count** are exclusive with option **-waveform\_interval** in this command.

The following is the table of options supported in each power analysis mode.

|                            |     |          |            |                         |  |
|----------------------------|-----|----------|------------|-------------------------|--|
|                            |     |          |            | time_based              |  |
| leakage_variation          |     | averaged | time_based | (RTL or zero delay vcd) |  |
|                            |     |          |            |                         |  |
| static_leakage_only        |     | Yes      | No         | No                      |  |
|                            | No  |          |            |                         |  |
|                            |     |          |            |                         |  |
| variation_quantile         |     | No       | No         | No                      |  |
|                            | Yes |          |            |                         |  |
|                            |     |          |            |                         |  |
| waveform_interval          |     | No       | Yes        | No                      |  |
|                            | No  |          |            |                         |  |
|                            |     |          |            |                         |  |
| cycle_accurate_cycle_count |     | No       | No         | Yes                     |  |
|                            | No  |          |            |                         |  |
|                            |     |          |            |                         |  |
| cycle_accurate_clock       |     | No       | No         | Yes                     |  |
|                            | No  |          |            |                         |  |
|                            |     |          |            |                         |  |
| waveform_format            |     | No       | Yes        | Yes                     |  |
|                            | No  |          |            |                         |  |
|                            |     |          |            |                         |  |

|                     |     |     |     |  |
|---------------------|-----|-----|-----|--|
| waveform_output     | No  | Yes | Yes |  |
| No                  |     |     |     |  |
| include             | No  | Yes | Yes |  |
| No                  |     |     |     |  |
| include_groups      | No  | Yes | Yes |  |
| No                  |     |     |     |  |
| cells               | Yes | Yes | Yes |  |
| Yes                 |     |     |     |  |
| sdpd_tracking       | No  | Yes | Yes |  |
| No                  |     |     |     |  |
| sdpd_tracking_cells | No  | Yes | Yes |  |
| No                  |     |     |     |  |

When SDPD tracking is enabled, SDPD (state dependent and path dependent) switching activity for the specified cells will be tracked during power analysis. So later when write\_saif is performed, the result SAIF file will have the SDPD switching activities.

## EXAMPLES

The following example will cause PrimeTime PX to do time\_based power analysis. The time interval is 10 ns. The waveform output file will be my\_design.fsdb.

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -strip_path this_strip_path my_design.vcd
pt_shell> set_power_analysis_options -waveform_interval 10 -
waveform_output my_design
pt_shell> update_power
```

The following example will cause PrimeTime PX to do averaged power analysis. Only leakage power is calculated for fast turn-around time.

```
pt_shell> set power_analysis_mode averaged
pt_shell> set_power_analysis_options -static_leakage_only
pt_shell> update_power
```

The following example will cause PrimeTime PX to do time\_based power analysis for an RTL based design. The reference clock is *clk*.

```
pt_shell> set power_analysis_mode time_based
pt_shell> source name_mapping_info.tcl
pt_shell> read_vcd -rtl -strip_path this_strip_path my_rtl_design.vcd
pt_shell> set_power_analysis_options -cycle_accurate_clock clk
pt_shell> update_power
```

The following example will cause PrimeTime PX to do leakage power variation analysis. The variation quantile is 0.9.

```
pt_shell> set power_analysis_mode leakage_variation
pt_shell> set_power_analysis_options -variation_quantile quantile
pt_shell> update_power
```

## SEE ALSO

`power_analysis_mode(3)`,  
`report_power_analysis_options(2)`,  
`update_power(2)`,  
`report_power(2)`.

## **set\_power\_clock\_scaling**

Specify clock clock scaling for power analysis.

### **SYNTAX**

```
int set_power_clock_scaling
[-period period_value]
[-ratio ratio_value]
[clock_objects]
```

### **Data Types**

|                     |       |
|---------------------|-------|
| <i>period_value</i> | float |
| <i>ratio_value</i>  | float |

### **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.

Note that the scaling result is accumulated since the scaled switching activities in design are saved. 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 0.1, the 1st scaling results in the net's toggle rate to 0.1 (the original toggle rate 1.0 is replaced), the second scaling results in 0.01, and so on. To come back to the original toggle rates, you have to use the **reset\_switching\_activity** command and read the VCD or SAIF file again.

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.spf.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\_dерate**

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_dерate
 [-switching]
 [-internal]
 [-leakage]
 [-groups group_name_list]
 derate_value
 object_list
```

### **Data Types**

|                        |       |
|------------------------|-------|
| <i>group_name_list</i> | list  |
| <i>derate_value</i>    | float |
| <i>object_list</i>     | 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.

***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)
```

`set_power_derate`

1304

```
report_power_derate(2)
reset_power_derate(2)
```

## **set\_program\_options**

Sets options that affect the performance and capacity.

### **SYNTAX**

```
status set_program_options
[-enable_high_capacity]
[-disable_high_capacity]
[-enable_eco]
[-disable_eco]
[-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\_eco
  - Enables the ECO mode of the program. By default, ECO mode is disabled.
- disable\_eco
  - Disables the ECO mode of the program. Performance and capacity might be improved.
- 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 without affecting the quality of the timing analysis results.

#### ***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.

## ECO Mode

By default, the tool improves performance and capacity by removing unconnected cells during linking. However, since ECO commands might use those unconnected cells, you should enable ECO mode, which retains unconnected cells during linking.

## 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, and the **report\_timing** command shows graph-based analysis results.

The fast analysis mode and high capacity mode settings are preserved by the **save\_session** command; therefore, the **restore\_session** command always overrides the modes in the current session and restores the status of the modes that are saved.

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:

```
pt_shell> set sh_high_capacity_effort high
Information: The setting will be effective after next 'set_program_options' command
. (PTHC-001)
high

pt_shell> set_program_options -enable_high_capacity
Information: enabling high capacity analysis in 'high' effort..... (PTHC-001)
pt_shell> printvar sh_high_capacity_enabled
sh_high_capacity_enabled = "true"
```

The following command enables fast analysis mode:

```
pt_shell> set_program_options -enable_fast_analysis
Information: Fast analysis mode is enabled. (PTANA-002)

pt_shell> printvar sh_fast_analysis_mode_enabled
sh_fast_analysis_mode_enabled = "true"
```

## SEE ALSO

[pt\\_tmp\\_dir\(3\)](#)  
[sh\\_eco\\_enabled\(3\)](#)  
[sh\\_fast\\_analysis\\_mode\\_enabled\(3\)](#)

```
sh_high_capacity_effort(3)
sh_high_capacity_enabled(3)
```

set\_program\_options  
1308

## **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**

|                         |       |
|-------------------------|-------|
| <i>transition_value</i> | float |
| <i>object_list</i>      | 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**

|                          |       |
|--------------------------|-------|
| <i>pulse_width_value</i> | float |
| <i>object_list</i>       | 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  $\geq 0$ ).

*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  $\geq 0$ ). 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**

|                          |       |
|--------------------------|-------|
| <i>pulse_width_value</i> | float |
| <i>object_list</i>       | 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  $\geq 0$ ).

*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**

|                     |        |
|---------------------|--------|
| <i>class_name</i>   | string |
| <i>attr_name</i>    | string |
| <i>value</i>        | string |
| <i>object_names</i> | 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 | clk_to_output]
[-lib_cell lib_cell]
[-pin pin_name]
[-clock pin_name]
[-value parameter_value]
```

### **Data Types**

|                        |        |
|------------------------|--------|
| <i>lib_cell</i>        | string |
| <i>pin_name</i>        | string |
| <i>parameter_value</i> | float  |

### **ARGUMENTS**

```
-param setup | hold | clk_to_output
 Sets one of the following global parameters: setup, hold, 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**, or **clk\_to\_output**. These parameters specify the global parameter which is added to the arcs. To the QTM setup arc, the time is added to arrive at the final delay. To the QTM hold arc, the global hold 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**

|                    |        |
|--------------------|--------|
| <i>drive_type</i>  | string |
| <i>drive_value</i> | float  |
| <i>rtrans</i>      | float  |
| <i>ftrans</i>      | float  |
| <i>port_list</i>   | 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**

|                              |        |
|------------------------------|--------|
| <i>load_type</i>             | string |
| <i>multiplication_factor</i> | float  |
| <i>load_value</i>            | float  |
| <i>port_list</i>             | 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**

|                          |        |
|--------------------------|--------|
| <i>name</i>              | string |
| <i>trans_value</i>       | float  |
| <i>cap_value</i>         | float  |
| <i>wlm_name</i>          | string |
| <i>opcond_name</i>       | string |
| <i>process_value</i>     | float  |
| <i>voltage_value</i>     | float  |
| <i>temperature_value</i> | 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 multicornor, 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 wlml.

```
pt_shell> set_qtm_technology -wire_load_model wlml
```

## 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**

|                       |      |
|-----------------------|------|
| <i>separator_list</i> | list |
| <i>bus_name_list</i>  | list |
| <i>class_list</i>     | list |
| <i>regsub_list</i>    | list |
| <i>suffix_list</i>    | 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_po
rt}
 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. (UI
D-1067)
0
```

## SEE ALSO

`load_upf(2)`

## **set\_rail\_voltage**

Sets power rail voltage on cells.

### **SYNTAX**

```
int set_rail_voltage
[-rail_value rvalue]
[-rail_list rname_value_list]
[-min]
[-max]
[-dynamic_rail_value dynamic_component]
[-dynamic_rail_list name_value_list]
cell_list
```

### **Data Types**

|                          |       |
|--------------------------|-------|
| <i>rvalue</i>            | float |
| <i>rname_value_list</i>  | list  |
| <i>dynamic_component</i> | float |
| <i>name_value_list</i>   | list  |
| <i>cell_list</i>         | list  |

### **ARGUMENTS**

**-rail\_value *rvalue***

Sets voltage on single-rail cells. If you use this option, you cannot use the **-rail\_list** option because these options are mutually exclusive. Replace *rvalue* with a decimal or an integer value.

**-rail\_list *rname\_value\_list***

Sets voltages on individual rails of multirail cells. Replace *rname\_value\_list* with a list, which must have even number of elements. Odd elements are names of rails; even elements are voltage values. The rail names must match the definitions of rails in the library power\_supply section and in the library operating conditions. If you use this option, you cannot use the **-rail\_value** option because these options are mutually exclusive.

**-min**

Sets rail voltages for the minimum operating condition. If you do not specify either the **-min** or **-max** option, the tool assumes you are using both the **-min** and **-max** options.

**-max**

Sets rail voltages for the maximum operating condition. If you do not specify either the **-min** or **-max** option, the tool assumes you are using both the **-min** and **-max** options.

**-dynamic\_rail\_value *dynamic\_component***

Specifies what portion of *rvalue* is dynamic. You can only specify this option if the **-rail\_value** option is specified. If you attempt to specify this option with the **-rail\_list** option, an error message is issued.

```
-dynamic_rail_list name_value_list
 Specifies the dynamic portion of each rail voltage listed with the -rail_list
 option. The list of dynamic component values must match the entries in the
 rname_value_list.
```

```
cell_list
 Specifies a list of cells on which to set rail voltages. Replace cell_list
 with the collection of cells you want.
```

## DESCRIPTION

Sets power rail voltage on cells. This command overrides voltages specified in operating conditions. It can be applied to both leaf cells and hierarchical blocks. If used on hierarchical cells, all cells in the block must come from the same library, unless another **set\_rail\_voltage** or **set\_operating\_conditions** command has been set on them.

It should be noted that since *-min* and *-max* refer to operating conditions, the *-max* voltage is typically lower than the *-min* voltage.

The voltages are passed to delay equations in the library. Ground rail voltages cannot be changed; the tool assumes zero voltage.

In a typical multi-voltage flow you can set operating conditions on hierarchical blocks corresponding to voltage islands. You can then use a power network analysis tool to calculate the IR drop in power networks; subtract power and ground voltages on cells; and, by using the **set\_rail\_voltage** command, annotate them on important leaf cells or blocks.

The *-dynamic\_rail\_value* or *-dynamic\_rail\_list* option can be used to specify the dynamic component of IR drop. In this case, the *-rail\_value* or *-rail\_list* option can be used to specify the total voltage on a cell and the dynamic options can specify what portion of that voltage is dynamic.

Care should be taken when specifying the dynamic component of the rail voltage for the max operating condition. In this case, the dynamic component should be negative. This ensures that the static component of rail voltage is not less than the total minimum rail voltage (corresponding to max operating condition).

If Clock Reconvergence Pessimism Removal (CRPR) is enabled and dynamic rail voltages are specified, the Dynamic CRPR mode is turned on. In this mode only CRPR is computed using clock totals considering the dynamic component of rail voltage if the path is zero-cycle. A zero-cycle path is one in which the same clock edge both launches and captures.

Note that the operating condition (or rail voltage) that is set on the lower levels of the hierarchy (or leaf cell) overrides the operating condition (or rail voltage) on the higher levels of hierarchy.

## EXAMPLES

The following example sets the operating conditions named *OC1* and *OC2* on the block named *h1* and then overrides the voltage on the cell named *u3* in *h1*.

```
pt_shell> set_operating_conditions -min OC1 -max OC2 \
 -object_list [get_cells {h1}]
1

pt_shell> set_rail_voltage -min -rail_value 1.4 [get_cells {h1/u3}]
1

pt_shell> set_rail_voltage -max -rail_value 1.1 [get_cells {h1/u3}]
1
```

To specify a dynamic component of 0.2 for the example above, you should use the following commands (Note the negative value for the max operating condition):

```
pt_shell> set_rail_voltage -min -rail_value 1.4 -
 dynamic_rail_value 0.2 [get_cells {h1/u3}]
1

pt_shell> set_rail_voltage -max -rail_value 1.1 -dynamic_rail_value -
 0.2 [get_cells {h1/u3}]
1
```

## SEE ALSO

`remove_rail_voltage(2)`  
`set_operating_conditions(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]
[power_net]
[-reset]
```

### **Data Types**

|                   |                |
|-------------------|----------------|
| <i>ports</i>      | list           |
| <i>ground_net</i> | upf_supply_net |
| <i>power_net</i>  | upf_supply_net |

### **ARGUMENTS**

**-object\_list *objects***  
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.

**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.

**-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.

### **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**

```
int 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]
[-elements object_list]
retention_name
```

### **Data Types**

|                        |        |
|------------------------|--------|
| <i>domain_name</i>     | string |
| <i>supply_net_name</i> | string |
| <i>supply_set_name</i> | string |
| <i>object_list</i>     | list   |
| <i>retention_name</i>  | 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.

-elements object_list
 Specifies the objects that this UPF retention strategy is applied to. 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.

## 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 shutdown_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**

```
help UPF
report_power_pin_info(2),
set_retention_control(2),
create_supply_net(2),
create_supply_set(2),
create_power_domain(2)
```

## **set\_retention\_control**

Defines the UPF retention control signals for the defined UPF retention strategy.

### **SYNTAX**

```
int set_retention_control
retention_strategy_name
-domain power_domain
-save_signal save_signal_and_save_sense
-restore_signal restore_signal_and_restore_sense
```

### **Data Types**

|                                  |                 |
|----------------------------------|-----------------|
| retention_strategy               | string          |
| power_domain                     | string          |
| save_signal_and_save_sense       | string high low |
| restore_signal_and_restore_sense | string high low |

### **ARGUMENTS**

retention\_strategy\_name  
Specifies the UPF retention strategy name. The retention strategy should have already been defined using **set\_retention** command

-domain power\_domain  
Specifies the power domain name that this UPF retention strategy that is applied to.

-save\_signal save\_signal\_net\_sense  
Specifies the save signal net and the save signal sense for the retention cells under this retention strategy.

-restore\_signal restore\_signal\_and\_restore\_sense  
Specifies the restore signal net and the restore sense for the retention cells under this retention strategy.

### **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_stra \
tegy]] \
}
```

## 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**

```
int set_rtl_to_gate_name
-rtl rtl_name
-gate gate_name
-substitute {substitute}
-inverted
```

### **Data Types**

|                  |        |
|------------------|--------|
| <i>rtl_name</i>  | string |
| <i>gate_name</i> | 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**

```
string set_sense
[-type clock | data]
[-positive]
[-negative]
[-stop_propagation]
[-pulse pulse_type]
[-clocks clock_list]
object_list
```

### **Data Types**

|                    |        |
|--------------------|--------|
| <i>pulse_type</i>  | string |
| <i>clock_list</i>  | list   |
| <i>object_list</i> | 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.

**-positive**

Specifies positive unateness applied 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**

Specifies negative unateness applied 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.

**-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

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.

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
```

## SEE ALSO

`remove_sense(2)`

## **set\_setup\_hold\_pessimism\_reduction**

Set the optimization constraints for setup-hold pessimism reduction.

### **SYNTAX**

```
set_setup_hold_pessimism_reduction
-mode mode_name
[-setup_cutoff cutoff_slack]
[-hold_cutoff cutoff_slack]
```

### **Data Types**

|                    |        |
|--------------------|--------|
| mode_name          | string |
| setup_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 cutoff\_slack  
Specifies the cutoff slack of the MAX path for SHPR optimization. Must be a float value. The default value is negative INFINITY.

-hold\_cutoff cutoff\_slack  
Specifies the cutoff slack of the MIN path for SHPR optimization. Must be a float value. The default value 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 value 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**

```
int set_si_aggressor_exclusion
[-number_of_active_aggressors n]
[-rise]
[-fall]
anets
```

### **Data Types**

*anets*      list

### **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 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**

```
int set_si_delay_analysis
[-ignore_arrival inets]
[-exclude]
[-victims vnets]
[-aggressors anets]
[-rise]
[-fall]
[-min]
[-max]
```

### **Data Types**

|              |      |
|--------------|------|
| <i>rnets</i> | list |
| <i>inets</i> | list |
| <i>vnets</i> | list |
| <i>anets</i> | 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.

**-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\_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**

|                    |       |
|--------------------|-------|
| <i>res_value</i>   | float |
| <i>object_list</i> | 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\_switching\_activity**

Sets switching activity annotation on the selected nets, pins, ports, and cells of the current design.

### **SYNTAX**

```
int 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**

|                           |        |
|---------------------------|--------|
| <i>state_condition</i>    | string |
| <i>path_sources</i>       | list   |
| <i>toggle_rate</i>        | float  |
| <i>derate_value</i>       | float  |
| <i>glitch_rate</i>        | float  |
| <i>static_probability</i> | float  |
| <i>rise_ratio</i>         | float  |
| <i>period_value</i>       | float  |
| <i>clock</i>              | string |
| <i>object_type_list</i>   | list   |
| <i>clock_list</i>         | list   |
| <i>object_list</i>        | 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_derrate 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 = 0.5 and the toggle rate on the clock gating output is  $0.1 * 0.5 = 0.05$ .

## SEE ALSO

```

create_clock(2)
get_switching_activity(2)
read_saif(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)

```

[set\\_switching\\_activity](#)

## **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**

Sets derating factors for either the current design or a specified list of instances (cells, library cells, or nets).

### **SYNTAX**

```
int set_timing_derate
-early | -late
[-rise] [-fall]
[-clock] [-data]
[-cell_delay] [-cell_check] [-net_delay]
[-static] [-dynamic]
[-scalar | -variation | -aocvm_guardband | -pocvm_guardband]
[-pocvm_coefficient_scale_factor]
[-increment]
derate_value
object_list
```

### **Data Type**

|              |       |
|--------------|-------|
| derate_value | float |
| object_list  | list  |

### **ARGUMENTS**

-early  
Indicates that the *derate\_value* specified should be applied to early delays (shortest paths). This option and the *-late* option are mutually exclusive.

-late  
Indicates that the *derate\_value* specified should be applied to late delays (longest paths). This option and the *-early* option are mutually exclusive.

-rise  
Indicates that the *derate\_value* specified should be applied to rise delays.

-fall  
Indicates that the *derate\_value* specified should be applied to fall delays.

-clock  
Indicates that the specified derating factor should apply only to clock paths.

-data  
Indicates that the specified derating factors should apply only to data paths.

-net\_delay  
Indicates that the specified derating factor should apply to net delays. This option cannot be specified with the *-cell\_check* option.

**-cell\_delay**  
Indicates that the specified derating factor should apply to cell delays. This option cannot be specified with the **-cell\_check** option.

**-cell\_check**  
Indicates that the specified derating factor should apply only to cell timing checks. If this option is specified, the derate value set with the **-early** option is applied to hold and removal timing checks and the derate value set with the **-late** option is applied to setup and recovery timing checks. This option cannot be specified with any of the following options: **-clock**, **-data**, **-cell\_delay**, **-net\_delay**, **-aocvm\_guardband** and **-pocvm\_guardband**.

**-static**  
Indicates that the specified derating factor should apply to the nondelta portion of net delays only. This option requires that the **-net\_delay** option is specified.

**-dynamic**  
Indicates that the specified derating factor should apply to the dynamic component of delays only. This option requires that the **-net\_delay** option is specified. This option cannot be specified with the **-aocvm\_guardband** option or **-pocvm\_guardband** option.

**-scalar**  
Indicates that the specified derating factor should apply to deterministic delays only.

**-variation**  
Indicates that the specified derating factor should apply to statistical delays only.

**-aocvm\_guardband**  
This option is only applicable in an advanced on-chip variation (OCV) context. In an advanced OCV context, the derate factor that is applied to an arc is a product of the guard-band derate factor and the advanced OCV derate factor. This option cannot be specified with any of the following options: **-scalar**, **-variation**, **-dynamic**, or **-cell\_check**.

**-pocvm\_guardband**  
This option is only applicable in a parametric on-chip variation (OCV) context. In a parametric OCV context, the derate factor that is applied to an arc is a product of the guard-band derate factor and the parametric OCV derate factor corresponding to a distance-based parametric OCV table, if specified. The parametric OCV factor applied to the standard deviation is also multiplied by the guardband factor. This option cannot be specified with any of the following options: **-scalar**, **-variation**, **-dynamic**, or **-cell\_check**.

**-pocvm\_coefficient\_scale\_factor**  
This option is only applicable in a parametric on-chip variation (OCV) context. In a parametric OCV context, the coefficient applied to an arc is a product of the parametric on-chip variation (POCV) coefficient and the parametric on-chip variation (POCV) coefficient scale factor.

**-increment**  
Indicates that the specified derating factor is an incremental derate factor

and should be added to the base derate factor to compute the total derate for an object. This option can not be specified with the `-aocvm_guardband` option.

**derate\_value**

Specifies the timing derate value that are applied to the specified delays as a scalar multiplicative factor.

**object\_list**

Specifies current design or a list of cells, library cells, or nets to which the specified derate factor are applied.

## DESCRIPTION

Sets derating factors for either the current design (if no object list is specified) or a set of cells, library cells, or nets in the current design.

Timing derating factors affect delay values shown in timing reports. Longest path delays (for example, launching paths for setup checks or capturing paths for hold checks) are multiplied by the derate value set using the `-late` option, and shortest paths (for example, capturing paths for setup checks or launching paths for hold checks) are multiplied by the derate values set using the `-early` option. If derating factors are not specified, then a value of 1.0 is assumed.

The intended usage mode of this command is that you first set global or default derates on the design and then you can use the `object_list` option to set specific derate values for cells, library cells, or nets in the design. To set derates globally on a design simply issue the command with no object list specified. For example, to set an early derate factor of value 0.95 on all cell and net delays in the design, use:

```
set_timing_derate -early 0.95
```

If only the `-net_delay` option is specified then the derate factor supplied will be applied to all net delays in the design (if no `object_list` is specified). If an `object_list` is specified, the derate factor is applied to each net and/or to every net in each hierarchical cell in the `object_list`.

If only the `-cell_delay` option is specified, the derate factor supplied is applied to either all cell delays in the design (if no `object_list` is specified) or to each delay (apart from timing check delays) on each cell or library cell specified in the `object_list`.

If neither the `-net_delay`, `-cell_delay`, nor `object_list` option is specified with the command, it is assumed that the derate factor is to be applied to all cell and net delays in the design.

If `-net_delay` is specified with the command and neither `-static` nor `-dynamic` has been specified, it is assumed that the derate factor is to be applied to both the dynamic and nondynamic portions of net delays in the design.

If only the `-clock` option is specified, the derate factor supplied is only applied to delays that are in clock paths. If only the `-data` option is specified, the derate factor supplied is only applied to delays that are in data paths. If neither of the above options is specified, it is assumed that the derate factor is to be applied to

both clock and data paths.

If neither the *-scalar* nor *-variation* option is specified, it is assumed that the derate factor is to be applied to both deterministic and variation delays.

If you omit the *-net\_delay* or *-cell\_delay* option from the command, PrimeTime automatically determines the intended scope of the command from the types of objects in the object list. For example,

```
pt_shell> set_timing_derate -late 1.1 [get_nets n*]
```

Warning: Implicitly setting the 'net\_delay' option for the *set\_timing\_derate* command. (UIITE-437)

If the collection contains nets, PrimeTime infers the *-net\_delay* option. If the collection contains cells, PrimeTime infers the *-cell\_delay* option. If the collection contains both nets and cells, PrimeTime infers both.

The *-cell\_check* option applies the derate value to the setup and hold time requirements of cells: cell setup and cell recovery times for late derating or cell hold and cell removal times for early derating. These time values are timing constraints, not delays. The derate factor applies to the cell instances or library cells specified in the object list, or to all timing checks in the design if no object list is provided.

Note that the following commands set the derating factors for all cell and net delays:

```
pt_shell> set_timing_derate -late 1.3
pt_shell> set_timing_derate -early 0.8
```

whereas the following commands set the derating factors for all cell setup time and cell hold time constraints:

```
pt_shell> set_timing_derate -late 1.3 -cell_check
pt_shell> set_timing_derate -early 0.8 -cell_check
```

If the **timing\_use\_constraint\_derates\_for\_pulse\_checks** variable is set to *true*, the *derate\_value* value specified by the *-cell\_check -late* options also applies to the min-pulse-width and min-period constraints. In this case, the *-fall* option applies to min-pulse-width checks for low pulses, the *-rise* option applies to min-pulse-width checks for high pulses. The greater the *derate\_value* from the *-rise* or *-fall* option applies to min-period checks.

The *object\_list* option can be used to set specific derate factors on instances (cells or library cells) in the design. On each instance the *-cell\_delay*, *-cell\_check*, *-clock*, and *-data* options can be used in the same way as outlined above to specify exactly how the derate factors should be applied to each instance in the object list.

If no options are specified and the object list does not contain hierarchical cells, it is assumed that the *-cell\_delay*, *-net\_delay*, *-clock*, and *-data* options are *true*.

If no options are specified and the object list contains hierarchical cells, the *-cell\_delay*, *-net\_delay* or *-cell\_check* option must be specified.

If the `-net_delay` option is specified and the `object_list` contains any hierarchical cells, all nets within that hierarchical cell and also all nets of all lower hierarchical cells have that derate factor set on them.

When applying derate factors on multiple object types that apply to a given arc, then the following priority (in decreasing order of precedence) is used to determine the derate that applies to the arc:

- 1) leaf cell derate
- 2) library cell derate
- 3) hierarchical cell derate
- 4) design derate

Therefore, when derate factors are being applied for a cell, PrimeTime first checks if they have been set for the cell itself (leaf cell). It then checks for the library cell. It then checks if it has been set for any of the cells' hierarchical parents. If no derate factors are found, it uses the factors set globally on the design.

Note that the precedence rules above were different in releases of PrimeTime prior to version D-2010.06, where library cell derates had higher precedence than hierarchical cell derates. To use the old behavior, set the **timing\_derate\_precedence\_compatibility** variable to `true`.

The derate value specified using the `f-increment` option, is applied to both OCV and advanced OCV base derates. For a given object, the base derate is first computed and then the incremental derate is added on to determine the total derate factor. . The incremental derate factors adhere to the same inheritance, precedence and override rules followed by base derates. If the total derate factor after addition of incremental derate is negative, the tool will clip it to 0.0 and an error message will be issued. If no base derate exists for an object, a derate value of 1.0 will be assumed and incremental derate will be added to compute the total derate.

The `object_list` option can also be used to set specific derate factors on particular nets in the design. A derate factor set on a specific net has precedence over net derate factors set globally. If a derate factor is set on a portion of a hierarchical net, all portions of that hierarchical net also have that derate factor set and a warning is issued indicating this fact. Any values previously set on this hierarchical net is overwritten.

PrimeTime only inherits derates onto global nets from the parent cell of the highest hierarchical level of the global net. This means that derates are not inherited onto global nets from parent cells of any lower hierarchical levels. This convention resolves inconsistent application of derates onto global nets.

To set derating values back to the default (derate factor of 1.0 for every instance in the design), use the **reset\_timing\_derate** command.

For delays an early factor less than 1 and late factor more than 1, add pessimism to the slack calculation. However, constraint derating is applied directly to the value of constraint. For example, a factor of less than 1 for hold means making hold time smaller (for positive hold time) thus giving less conservative slack. A factor greater than 1 for setup means making setup time larger (for positive setup time) thus more conservative slack.

The following equation is used to apply derates to both positive and negative delays (in all cases except the corner case when the derate factor is specified as 0.0, which is typically done for investigative purposes):

```
• derated_delay = delay + ((derate_factor - 1.0)*ABS(delay))
```

This equation ensures that negative delays are correctly made more conservative by the application of derates. When the derate factor is set to 0.0 (typically used for investigative purposes), the derated delay is always 0.0 irrespective of whether the delays are positive or negative.

If signal integrity is enabled, net delays might also include delta delay resulting from crosstalk interaction between nets. Because minimum delta delays are negative, net derate factors are applied separately to the net\_delay (without any delta delay component) and delta delay. Both derated values are subsequently summed to give the total derate net delay.

In addition, if the **si\_use\_driving\_cell\_derate\_for\_delta\_delay** variable is set to true, the delta delay specified in the previous paragraph is derated using the relevant derate factor from the cell driving that net.

If the operating condition analysis type was previously set to *single*, executing the **set\_timing\_derate** command automatically switches to the *on\_chip\_variation* analysis type. This is equivalent to executing **set\_operating\_conditions -analysis\_type on\_chip\_variation** command. Otherwise, the analysis modes of *bc\_wc* or *on\_chip\_variation* remain unchanged.

If only global derate factors are specified, their values can be shown using the **report\_design** command. This also indicates what analysis mode PrimeTime is in.

To report all derate factors set for the design or for any cell, library cell, or net in the design, use the **report\_timing\_derate** command.

In an advanced on-chip variation (OCV) analysis static delay derates are ignored. For more information, see the *timing\_aocvm\_enable\_analysis* man page.

PrimeTime version V-2003.12 does not support using different timing derates for different operating conditions just like other tools, such as Design Compiler. Because of this, the tool does not accept per-operating condition derates read in from scripts or database files.

## EXAMPLES

The following example sets an early (shortest path) derate factor of 0.7 and a late (longest path) derate factor of 1.0 globally on all cell and net delays the design:

```
pt_shell> set_timing_derate -early 0.70
pt_shell> set_timing_derate -late 1.0
```

The following example sets an early derate factor of 0.8 for the cell delay of cell

U1:

```
pt_shell> set_timing_derate -early 0.8 -cell_delay [get_cells U1]
```

The following example sets a late derate factor of 1.1 on all instances of library cell IV in the library MY\_LIB:

```
pt_shell> set_timing_derate -late 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 late derate factor for the instance "inv" only:

```
pt_shell> set_timing_derate -late 1.05 [get_cells inv]
```

The following example illustrates how to set a derate factor on all cells and nets (including sub-hierarchies) within the hierarchy top/H1:

```
pt_shell> set_timing_derate -cell_delay -net_delay -late 1.05 [get_cells top/H1]
```

The following shows how to set an early derate factor of 0.7 on a specific net 'n1':

```
pt_shell> set_timing_derate -net_delay -early 0.7 [get_nets n1]
```

The following example sets a late derate factor of 1.15 (= 1.11 + 0.04) and early derate factor of 0.88 (= 0.91 + (-0.03)) for the cell delay of cell u1/u252:

```
pt_shell> set_timing_derate -late 1.09
pt_shell> set_timing_derate -early 0.91
pt_shell> set_timing_derate -late 1.11 [get_cells u1/u252]
pt_shell> set_timing_derate -increment -late 0.04 [get_cells u1/u252]
pt_shell> set_timing_derate -increment -early -0.03 [get_cells u1/u252]
```

## SEE ALSO

```
report_design(2)
report_timing(2)
report_timing_derate(2)
reset_timing_derate(2)
si_use_driving_cell_derate_for_delta_delay(3)
set_operating_conditions(2)
timing_derate_precedence_compatibility(3)
```

## **set\_units**

Checks the specified units with the main library units. The command fails if the units specified do not match the main library units.

### **SYNTAX**

```
int 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**

|                             |       |
|-----------------------------|-------|
| <i>time_in_sec</i>          | float |
| <i>capacitance_in_farad</i> | float |
| <i>current_in_ampere</i>    | float |
| <i>voltage_in_volt</i>      | float |
| <i>resistance_in_ohm</i>    | float |
| <i>power_in_watt</i>        | 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 units specified with the main library units. This command can only be used after the design has been linked. The main library units are the design units. The **set\_units** command does not allow setting of design units, but is intended to prevent 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 the **update\_power**, **read\_vcd**, **read\_saif**, and **set\_switching\_activity** commands. You can check all units using 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.

## 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

```
report_units(2)
read_sdc(2)
write_sdc(2)
write_script(2)
```

## **set\_unix\_variable**

This is a synonym for the **setenv** command.

### **SEE ALSO**

**getenv(2)**, **printenv(2)**, **printvar(2)**, **set(2)**, **setenv(2)**, **sh(2)**, **unset(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**

|                    |        |
|--------------------|--------|
| <i>class_name</i>  | string |
| <i>object_spec</i> | list   |
| <i>attr_name</i>   | string |
| <i>value</i>       | 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\_voltage**

Applies an operating voltage on a list of power nets or pg pins.

### **SYNTAX**

```
int set_voltage
[-min min_case_voltage]
[-min_dynamic dynamic_min_case_value]
[-dynamic dynamic_max_case_value]
[-object_list list_of_power_nets]
[-cell cell]
[-pg_pin_name pg_pin]
max_case_voltage
```

### **Data Types**

|                               |        |
|-------------------------------|--------|
| <i>min_case_voltage</i>       | float  |
| <i>dynamic_min_case_value</i> | float  |
| <i>dynamic_max_case_value</i> | float  |
| <i>list_of_power_nets</i>     | list   |
| <i>cell</i>                   | list   |
| <i>pg_pin</i>                 | string |
| <i>max_case_voltage</i>       | float  |

### **ARGUMENTS**

- min *min\_case\_voltage*  
Specifies the operating voltage for the minimum delay (fastest) case. This is typically a larger voltage value.
- min\_dynamic *dynamic\_min\_case\_value*  
Specifies a portion of the voltage changes that varies with frequency significantly higher than several clock periods. The dynamic portion is considered during capacitive coupling analysis and various pessimism removal optimizations.
- dynamic *dynamic\_max\_case\_value*  
Specifies a portion of the voltage changes that varies with frequency significantly higher than several clock periods. The dynamic portion is considered during capacitive coupling analysis and various pessimism removal optimizations.
- object\_list *list\_of\_power\_nets*  
Specifies the list of power nets that have the operating voltages specified in this command.
- cell *cell*  
Use this option together with the **-pg\_pin\_name** option to specify the power or ground pin of cell instance for IR drop annotation.
- pg\_pin\_name *pg\_pin*  
Use this option together with the **-cell** option to specify power or ground pin

of cell instance for IR drop annotation.

**max\_case\_voltage**

Specifies the operating voltage for the maximum delay (slowest) case. This is typically a smaller voltage value.

## DESCRIPTION

Defines the operating voltage on the power nets or pins so that the parts of the design powered by these power nets are timed and optimized at the specified voltage. If you do not specify any operating voltage for a power net, the part of the design connected by this power net continues to be timed and optimized based on the available operating condition settings.

If you specify voltage ranges for a power net, the operating voltages of the power net must fall within one of the ranges specified in the voltage ranges. Furthermore, the *min\_case\_voltage* must fall within the same range as the *max\_case\_voltage*.

To see the operating voltages of power nets, use the **report\_power\_net\_info** command.

After you define the operating voltage of a power net, it can be overridden only with the **set\_voltage** command. It can also be cleared using the **reset\_design** command.

When you set the voltage on UPF supply nets, the voltage is set only on the segments connected to the segment in **-object\_list** at the time of executing **set\_voltage**. Therefore, it is important to load the complete UPF description of the power network before issuing **set\_voltage**. To check whether any power net segment has missing voltage, use the **check\_timing -override** command.

## EXAMPLES

The following example shows a typical usage of the **set\_voltage** command:

```
prompt> create_power_domain TOP
1
prompt> create_supply_net VDD -domain TOP
1
prompt> create_supply_net VSS -domain TOP
1
prompt> set_domain_supply_net TOP -primary_power_net VDD \
 -primary_ground_net VSS
1
prompt> set_voltage 0.88 -min 1.188 -cell I1 -pg_pin_name PWR
1
prompt> set_voltage 0.04 -min -0.05 -cell I1 -pg_pin_name GND
1
```

## SEE ALSO

**check\_timing(2)**  
**create\_power\_domain(2)**  
**create\_supply\_net(2)**  
**report\_power\_domain(2)**

**set\_voltage**

```
report_power_pin_info(2)
```

set\_voltage  
1389

## **set\_waveform\_integrity\_analysis**

Sets static or dynamic constraints for waveform integrity analysis.

### **SYNTAX**

```
status set_waveform_integrity_analysis
[-static]
[-dynamic]
constraint_value
[object_list]
```

### **Data Types**

|                         |       |
|-------------------------|-------|
| <i>constraint_value</i> | float |
| <i>object_list</i>      | list  |

### **ARGUMENTS**

|                         |                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>-static</i>          | Applies the constraint to static waveform integrity analysis.                                                                                                                      |
| <i>-dynamic</i>         | Applies the constraint to dynamic waveform integrity analysis.                                                                                                                     |
| <i>constraint_value</i> | Specifies the constraint limit (Value $\geq 0$ ). For static analysis, this should be in library time unit. For dynamic analysis, this should be percentage of the supply voltage. |
| <i>object_list</i>      | Specifies a list of cell input pins.                                                                                                                                               |

### **DESCRIPTION**

Specifies constraint limit for waveform integrity analysis. For static analysis, the constraint value is the constraint limit for waveform distortion. If the distorted waveform from the advanced waveform propagation has larger distortion metric than the constraint value, it is reported by **report\_constraint -waveform\_analysis static** command. The distortion metric is calculated in the max time difference between the distorted waveform and its concave bounding waveform at the same voltage level. Thus, the constraint value for static analysis should be in library time unit. Note that the static constraint can be set only to the input pins of synchronous elements of the design. Since it requires the waveform, the advanced waveform propagation should be enabled for the static waveform integrity analysis.

For dynamic analysis, the constraint value is the constraint limit for the aggressor bumps. If the sum of all active aggressors to this input pin is larger than the constraint limit, it is reported by **report\_constraint -waveform\_analysis dynamic**. Thus, the constraint value for dynamic analysis should be in percentage of the supply voltage. Note that the dynamic constraint can be set to any input pins of the design, and available only when the signal integrity analysis is enabled.

The constraint value is applied to the specified cell input pins if the list of cell input pins are provided. If there is no list of cell input pins, the constraint value is applied as the design-level constraints. When both design-level and pin-level constraints are specified, the most restrictive constraint is used.

## EXAMPLES

The following example sets a static constraint of 0.3 library time unit on synchronous input pin DFF/CLK.

```
pt_shell> set_waveform_integrity_analysis -static 0.3 [get_pins DFF/CLK]
```

The following example sets a dynamic constraint of 0.5 on input pin mycell/A.

```
pt_shell> set_waveform_integrity_analysis -dynamic 0.5 [get_pins mycell/A]
```

## SEE ALSO

```
remove_waveform_integrity_analysis(2)
report_waveform_integrity_analysis(2)
report_constraint(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**

|                    |        |
|--------------------|--------|
| <i>model_name</i>  | string |
| <i>lib_spec</i>    | string |
| <i>object_list</i> | 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**

|                             |        |
|-----------------------------|--------|
| <i>lib_spec</i>             | string |
| <i>selection_group_name</i> | string |
| <i>object_list</i>          | 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

**set\_wire\_load\_selection\_group**

**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)**, **unsetenv(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)`.

## **sh\_list\_key\_bindings**

Displays all the key bindings and edit mode of current shell session.

### **SYNTAX**

```
string sh_list_key_bindings
[-nosplit]
```

### **ARGUMENTS**

**-nosplit** Prevents line splitting when column fields overflow.

### **DESCRIPTION**

The **sh\_list\_key\_bindings** command displays current key bindings and the edit mode. To change the edit mode, set the **sh\_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 you hold the Ctrl key while pressing the K key.

The text `META+K` is read as 'Meta+K' and describes the character produced when you hold down the Meta key while pressing the K key. 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 or Alt key, or another key working as a Meta key, the identical keystroke can be generated by pressing 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)`

## **sim\_analyze\_clock\_network**

Analyzes a clock network with a SPICE simulator.

### **SYNTAX**

```
int sim_analyze_clock_network
-from from_pins
[-from_rise_slew rise_slew]
[-from_fall_slew fall_slew]
[-output model_file_name]
[-to to_pins]
[-use_probe]
```

### **Data Types**

|                        |            |
|------------------------|------------|
| <i>from_pins</i>       | collection |
| <i>rise_slew</i>       | float      |
| <i>fall_slew</i>       | float      |
| <i>model_file_name</i> | string     |
| <i>to_pins</i>         | collection |

### **ARGUMENTS**

-from *from\_pins*  
Specifies a collection of one root pin or port of the clock network. clock network is extracted by tracing from the root until sequential loads are reached. Clock gating cells are included in the clock network. It is essential for you to choose the appropriate root for clock network to satisfy their needs.

-from\_rise\_slew *rise\_slew*  
Specifies the rise slew at the clock root node in main library units. If specified, it is used for creating input stimulus. Otherwise, slew from PrimeTime is used.

-from\_fall\_slew *fall\_slew*  
Specifies the fall slew at the clock root node in main library units. If specified, it is used for creating input stimulus. Otherwise, slew from PrimeTime is used.

-output *model\_file\_name*  
Specifies the file name in which the results of the simulations are written. It is written in Synopsys Design Constraint format using the **set\_annotated\_delay**, **set\_annotated\_transition** and **set\_disable\_timing** commands. This file is the clock mesh model that can be used for multiple runs on the same design of PrimeTime if clock network has not changed between runs.

-to *to\_pins*  
Specifies the collection of pins, when reached, stops the clock network tracing. If one of the load pins of the net is included, other load pins of the net are also automatically included to stop clock network tracing. This

option helps control the simulation and back annotation of the clock network logic that is being analyzed.

#### -use\_probe

Adds probe points to the SPICE simulation for all pins in the clock network.

## DESCRIPTION

This command extracts the clock network from the specified clock root and invokes the simulator specified by the **sim\_setup\_simulator** command. The purpose of the command is to create a SPICE deck, link to simulation environment, and back-annotate the results into PrimeTime. The SPICE deck generated is sensitized using the same rules as the **write\_spice\_deck** command. The input stimulus at the clock root is created based on the clock slew reaching at the root node. If you specify the rise and fall slews, they are used to create the input stimulus. Otherwise, the default rise and fall slews from PrimeTime are used. Only a single clock network can be traced with the specified *from\_pins* and *to\_pins*.

This command triggers an implicit **update\_timing** command to extract timing and sensitization information needed for generating SPICE deck of the clock network.

This command errors out when any of these features are enabled: crosstalk analysis, power analysis, variation analysis, multicore distributed analysis, HyperScale analysis.

The command also errors out when variable timing\_reduce\_multi\_drive\_net\_arcs is enabled.

To use the **sim\_analyze\_clock\_network** command, you must have a PrimeTime SI license.

## EXAMPLES

The following example analyzes the clock network starting from CLK port and creates a corresponding clock model in *clock\_model.tcl* file.

```
pt_shell> sim_analyze_clock_network -from [get_ports CLK] \
 -output clock_model.tcl
```

## SEE ALSO

**sim\_setup\_simulator**(2)  
**sim\_setup\_library**(2)  
**update\_timing**(2)  
**write\_spice\_deck**(2)

## **sim\_enable\_si\_correlation**

Enables signal integrity correlation mode for specific nets.

### **SYNTAX**

```
status sim_enable_si_correlation
net_list
```

### **Data Types**

*net\_list*      list

### **ARGUMENTS**

*net\_list*  
Specifies a list of nets for signal integrity SPICE correlation. The *net\_list* accepts a maximum of 1000 nets.

### **DESCRIPTION**

This command specifies nets for signal integrity SPICE correlation.

To use this command, you must

- Enable crosstalk analysis by setting **si\_enable\_analysis** to **true**.
- Have PrimeTime SI license
- Follow the recommended settings for SPICE correlation

### **EXAMPLES**

The following example enables signal integrity correlation on a net and validates coupled stage accuracy on the corresponding net arc.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects I1/Z]
pt_shell> update_timing -full
pt_shell> sim_validate_stage [get_timing_arc -from I1/Z -to I2/A]
```

### **SEE ALSO**

```
sim_setup_library(2)
sim_setup_simulator(2)
sim_validate_stage(2)
update_timing(2)
write_spice_deck(2)
si_enable_analysis(3)
```

## **sim\_setup\_library**

Sets up SPICE files for gate-level library.

### **SYNTAX**

```
status sim_setup_library
 -library library
 -sub_circuit dir
 -header file
 [-file_name_pattern pattern]
```

### **Data Types**

|                |            |
|----------------|------------|
| <i>library</i> | collection |
| <i>dir</i>     | string     |
| <i>file</i>    | string     |
| <i>pattern</i> | string     |

### **ARGUMENTS**

```
-library library
 Specifies the PrimeTime gate-level library for the SPICE setup.

-sub_circuit dir
 Specifies the directory that contains the subcircuit files (one file per
 cell) that are included in the simulation decks. Specifying a single file
 instead of a directory is allowed but might result in poor simulation
 performance.

-header file
 Specifies a file to be included at the beginning of the SPICE file. This file
 must contain a model file, corner instantiation, and simulator options. The
 file must not contain any measure statement or other circuit-specific
 statements. Generally, the SPICE options must be the same in all header files
 for each library. Temperature must not be present in the header files. It is
 automatically added to the SPICE deck by PrimeTime. By default, the power
 rail name is VDD (which uses the main library voltage), and the ground rail
 name is VSS (which uses 0 volts). To define rails other than VDD and VSS,
 specify rail votlage definiations in the header files.

-file_name_pattern pattern
 Restricts the files used for this library to those that follow the specified
 file name pattern. This option is helpful if the directory specified by the
 -sub_circuit dir option contains subcircuits for multiple libraries.
```

### **DESCRIPTION**

This command performs the setup for library like mapping the transistor models to the gate-level models.

## EXAMPLES

This example associates a specified library with a SPICE subcircuit directory and header file. The file pattern helps in associating the library cell names to my\_{lib\_cell\_name}.spc files in the subcircuit directory.

In this example, the IV170BQ library cell uses the my\_IV170BQ.spc file in the subcircuit directory to get its subcircuit.

```
pt_shell> sim_setup_library -library my_lib \
 -sub_circuit /path/gem/hspice \
 -header /path/gem/model_hspice \
 -file_name_pattern {my_%s.spc}
```

## SEE ALSO

[sim\\_analyze\\_clock\\_network\(2\)](#)  
[sim\\_setup\\_simulator\(2\)](#)

## **sim\_setup\_simulator**

Sets up the simulation environment.

### **SYNTAX**

```
status sim_setup_simulator
[-simulator sim_exec_path]
[-simulator_type hspice | nanosim | hsim]
[-sim_options option_string]
[-work_dir directory]
[-preserve failed | all | none]
```

### **Data Types**

|                      |        |
|----------------------|--------|
| <i>sim_exec_path</i> | string |
| <i>option_string</i> | string |
| <i>directory</i>     | string |

### **ARGUMENTS**

```
-simulator sim_exec_path
 Specifies the path to the simulation executable.

-simulator_type hspice | nanosim | hsim
 Specifies one of the following simulators
 • hspice (default) - HSPICE
 • nanosim - NanoSim
 • hsim - HSIM
 If you do not use this option, PrimeTime tries to derive the simulator type
 by using the -simulator option. If PrimeTime cannot derive the simulator
 type, the default simulator is HSPICE.

-sim_options option_string
 Specifies the default simulator options that are appended to the SPICE decks.

-work_dir directory
 Specifies the directory that stores temporary SPICE files. This option
 overrides any previous -work_dir options -- that is, there is only one for
 all libraries. The default directory is $pt_tmp_dir.

-preserve failed | all | none
 Specifies whether temporary files, such as SPICE decks, are kept on the disk.
 • all - Preserves all files.
 • none - Deletes all files.
 • failed (default) - Keeps the files only if simulation failed.
```

## **DESCRIPTION**

This command sets up the simulation environment. It specifies the simulator location, the type of simulator, and the working directory.

## **EXAMPLES**

The following example sets the simulator exec, simulator type, and the default options for the simulator.

```
pt_shell> sim_setup_simulator -simulator /global/apps/hspice/bin/hspice \
 -simulator_type hspice \
 -sim_options ".option ingold=2 statfl=1 acct=0 autostop brief runlvl=5"
```

The following example sets the work directory and marks all the temporary files to be preserved after simulation has finished.

```
pt_shell> sim_setup_simulator -work_dir [pwd] \
 -preserve all
```

## **SEE ALSO**

`sim_analyze_clock_network(2)`  
`sim_setup_library(2)`

## **sim\_setup\_spice\_deck**

Specifies setup options for SPICE deck generation.

### **SYNTAX**

```
status sim_setup_spice_deck
[-enable_clock_mesh]
[-use_pin_load]
```

### **ARGUMENTS**

#### **-enable\_clock\_mesh**

Enables the mesh model generation flow for performing clock network analysis using the **sim\_analyze\_clock\_network** command. Set this option to enable the clock model generation flow before reading the parasitics of the design. Otherwise, there is a possibility of a few large multidriven nets (mesh nets) being treated as ideal nets.

#### **-use\_pin\_load**

Replaces all the side-load cell instances by its input pin capacitance. The input pin capacitance used is nonlinear delay model (NLDM) pin capacitance of the lib cell specified in the library. CCS pin capacitances ( $c_1$ ,  $c_2$ ) are ignored and an equivalent pin cap is used in the SPICE deck. This option is beneficial in some cases because it improves the SPICE simulation runtime and also enables the SPICE simulations for load cells that do not have corresponding SPICE models. However, using this option results in loss of accuracy of SPICE simulation because the physical load cells are replaced by its equivalent pin cap model. This feature is also honored by the SPICE deck that is generated by the **write\_spice\_deck** command.

### **DESCRIPTION**

This command specifies setup options for writing out the SPICE deck.

### **EXAMPLES**

The following example enables clock mesh model generation flow.

```
pt_shell> sim_setup_spice_deck -enable_clock_mesh
```

The following example replaces all the side-loads in the SPICE deck with the input pin capacitance.

```
pt_shell> sim_setup_spice_deck -use_pin_load
```

### **SEE ALSO**

```
sim_analyze_clock_network(2)
write_spice_deck(2)
```

## **sim\_validate\_path**

Validates uncoupled PrimeTime accuracy compared with SPICE simulation.

### **SYNTAX**

```
status sim_validate_path
[-from start_pin_object]
[-to end_pin_object]
[-transition_time]
[-verbose]
path_objects
```

### **Data Types**

|                         |            |
|-------------------------|------------|
| <i>start_pin_object</i> | collection |
| <i>end_pin_object</i>   | collection |
| <i>path_objects</i>     | collection |

### **ARGUMENTS**

- from *start\_pin\_object*  
Specifies the start pin object for simulation.
- to *end\_pin\_object*  
Specifies the end pin object for simulation.
- transition\_time  
Shows the transition time comparison results. Without this option, only delays are compared.
- verbose  
Reports more details, such as the location of temporary files.
- path\_objects  
Specifies a collection of timing path objects to validate.

### **DESCRIPTION**

The **sim\_validate\_path** command simulates the specified path or path segment and compares the delay (and transition times) between PrimeTime and simulation.

If you specify a path by using the **-path\_type full\_clock\_expanded** option, the tool treats the path as a **full\_clock** path instead.

Before using this command, run the **sim\_validate\_setup** command to validate the SPICE simulation setup and environment. Incorrect setup of the simulation environment often causes correlation problems. Use a path-based analysis path for correlation purpose, and follow recommended settings for SPICE correlation as well.

This command works for uncoupled correlation purpose only. In designs with coupling, a **SIM-004** error is issued. Use the **sim\_validate\_stage** command for coupled

correlation.

To use the **sim\_validate\_path** command, you must have a PrimeTime SI license.

## EXAMPLES

The following example simulates a pba\_path and compares both delay and transition results with SPICE.

```
pt_shell> set pba_path [get_timing_path -pba_mode path]
pt_shell> sim_validate_path $pba_path -transition
```

The following example simulates a path segment of interest (U1/A to U2/Z), and compares delay results with SPICE.

```
pt_shell> set pba_path [get_timing_path -through U1/A \
 -through U2/Z -pba_mode path]
pt_shell> sim_validate_path $pba_path -from U1/A -to U2/Z
```

## SEE ALSO

[sim\\_setup\\_library\(2\)](#)  
[sim\\_setup\\_simulator\(2\)](#)  
[sim\\_validate\\_setup\(2\)](#)  
[sim\\_validate\\_stage\(2\)](#)

## **sim\_validate\_setup**

Validates SPICE simulation setup and environment.

### **SYNTAX**

```
status sim_validate_setup
-lib_cell library_cell
[-from from_pin]
[-to to_pin]
[-capacitance cap_value]
[-transition_time tran_value]
[-verbose]
```

### **Data Types**

|                     |            |
|---------------------|------------|
| <i>library_cell</i> | collection |
| <i>from_pin</i>     | string     |
| <i>to_pin</i>       | string     |
| <i>cap_value</i>    | float      |
| <i>tran_value</i>   | float      |

### **ARGUMENTS**

```
-lib_cell library_cell
 Specifies the library cell object to verify

-from from_pin
 Specifies the from pin of the library arc. The tool uses the library arc
 specified by the -from and -to options for setup verification.

-to to_pin
 Specifies the to pin of the library arc. The tool uses the library arc
 specified by the -from and -to options for setup verification.

-capacitance cap_value
 Specifies the output load to be driven by the library cell. The units are in
 PrimeTime main library units.

-transition_time tran_value
 Specifies the input transition time (slew) for the arc. A ramp or CCS standard
 pre-driver waveform (as per the library driver waveform) with this slew is
 generated at the corresponding input pin of this library cell. Both rise and
 fall will be simulated. The units are in PrimeTime main library units.

-verbose
 Shows more details such as location of temporary files.
```

### **DESCRIPTION**

Incorrect user setup of the simulation environment often causes problems. The goal of this command is to help you ensure that the PrimeTime libraries and SPICE

environment are matching. This command invokes the simulator specified by the **sim\_setup\_simulator** command on the given library cell arcs. The purpose is to verify that the basic characterization of a library matches the SPICE setup specified by the **sim\_setup\_\*** commands. To make the interface as simple as possible, only combinational cells are supported. If any error is detected in the SPICE run, FALSE is returned to the Tcl interpreter. Otherwise, the command displays the SPICE and PrimeTime computed delay and slew values, and returns TRUE.

This command checks for the presence of SPICE executable, SPICE netlist, SPICE model files, SPICE options, I/O rail settings, and correct units. It also checks the model license and compatibility of the SPICE version with the model.

This command does not replace Library checker or validation. The usage model is the other way around: Library has to be fully validated for accuracy first. This command assumes that the Liberty library is 100% correct and accurate, and therefore any PrimeTime and SPICE mismatch is due to incorrect simulation setup.

You can use this command only when there are no designs loaded in PrimeTime. Run this command after reading the libraries. The SPICE validation check is required one time for each library generally, for the first time the SPICE environment is setup in the design flow with the specific library.

To use the **sim\_validate\_setup** command, you must have a PrimeTime SI license.

## EXAMPLES

The example validates the simulation setup using a library arc from A to Y pins of IV170BQ lib cell with an output load of 48 library units and transition time of 0.2 units of library units.

```
pt_shell> sim_validate_setup -from A -to Y \
 -lib_cell [get_lib_cells $lib_name/IV170BQ] \
 -capacitance 48 -transition_time 0.2
```

## SEE ALSO

[sim\\_setup\\_library\(2\)](#)  
[sim\\_setup\\_simulator\(2\)](#)

## **sim\_validate\_stage**

Validates coupled PrimeTime accuracy compared with SPICE simulation.

### **SYNTAX**

```
int sim_validate_stage
[-analysis_type min_rise | max_rise | min_fall | max_fall]
[-transition_time]
[-verbose]
net_arc_objects
```

### **Data Types**

net\_arc\_objects      collection

### **ARGUMENTS**

```
-analysis_type min_rise | max_rise | min_fall | max_fall
 Reports the specified type of correlation, where "rise" or "fall" indicates
 a rising or falling transition on net arc load pin. If you do not specify
 this option, all four types of correlation are reported.

-transition_time
 Shows the transition time comparison results. Without this option, only
 delays are compared.

-verbose
 Reports more details, such as the location of temporary files.

net_arc_objects
 Specifies a collection of net timing_arc objects to validate.
```

### **DESCRIPTION**

The **sim\_validate\_stage** command simulates the specified stage and half and compares the delay (and transition times) between PrimeTime and simulation. For a net arc without receiver timing arc, stage correlation is reported.

Before using this command, run the **sim\_validate\_setup** command to validate the SPICE simulation setup and environment. Incorrect setup of the simulation environment often causes correlation problems. Run the **sim\_enable\_si\_correlation** command before the **update\_timing** command on nets of interest, and follow recommended settings for SPICE correlation.

This command works for coupled correlation purpose only. For uncoupled correlation, use the **sim\_validate\_path** command.

To use the **sim\_validate\_stage** command, you must have a PrimeTime SI license.

## EXAMPLES

The following example simulates one and half stage including U1/A to U1/Z cell arc, U1/Z to U2/A net arc, and U2/A to U2/Z cell arc. It compares both delay and transition results with SPICE for max\_rise.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects U1/Z]
pt_shell> update_timing -full
pt_shell> set net_arc [get_timing_arc -from U1/Z -to U2/A]
pt_shell> sim_validate_stage $net_arc -transition -analysis_type max_rise
```

## SEE ALSO

`sim_enable_si_correlation(2)`  
`sim_setup_library(2)`  
`sim_setup_simulator(2)`  
`sim_validate_path(2)`  
`sim_validate_setup(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**

|                  |        |
|------------------|--------|
| <i>lib_spec</i>  | list   |
| <i>cell_list</i> | list   |
| <i>lib_cell</i>  | 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.

## RESOLVING LIBRARY CELLS

If the *lib\_cell* option has been specified in base name only format, i.e. 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.

## 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)

size\_cell  
1420

## **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)`

`sizeof_collection`  
1422

## **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**

|                   |            |
|-------------------|------------|
| <i>collection</i> | collection |
| <i>criteria</i>   | 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
stringfile
```

### **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\_gui**

Starts the PrimeTime GUI.

### **SYNTAX**

```
string start_gui
[-file name_of_script_file]
[-no_windows]
```

### **Data Types**

*name\_of\_script\_file*      string

### **ARGUMENTS**

**-file** *name\_of\_script\_file*  
          Sources the given script file before the GUI starts.

**-no\_windows**  
          Starts the GUI without showing the default window.

### **DESCRIPTION**

This command starts the PrimeTime GUI from the pt\_shell prompt if the PrimeTime GUI has already been started.

### **EXAMPLES**

The following example starts the PrimeTime GUI.

```
pt_shell> start_gui
```

### **SEE ALSO**

`stop_gui(2)`

## **start\_hosts**

Start the hosts specified by the **set\_host\_options** command.

### **SYNTAX**

```
int start_hosts
[-timeout seconds]
[-min_hosts num_hosts]
```

### **Data Types**

|                |         |
|----------------|---------|
| <i>seconds</i> | integer |
| <i>number</i>  | 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 value of 21600s (6 hours) is used.

**-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.
```

## EXAMPLES

In the following example, the master process is running on the host named ptopt018 using a 64-bit binary and specifies five different sets of host options.

The first host options, named "my\_opts1", specifies 1 process with the same architecture as the master on the same host as the master and does not specify an upper limit on the number of cores to use; therefore, the default applies.

The second host options, named "my\_opts2", specifies 2 processes on the same host as the master, but explicitly mentions the name of the master's host and specifies to use a maximum of 2 cores per process.

The third host options, named "my\_opts3", specifies 4 processes (with the same architecture as the master) on the host named ptopt010 using "rsh" to connect to ptopt010 and specifying to use a maximum of 3 cores per process.

The fourth host options named "my\_opts4", specifies 5 processes (with the same architecture as the master) on an LSF farm, requesting 500MB of memory and 2 slots per compute server and specifying to use a maximum of 2 cores per process. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use hence the "-n 2 -R span\[ptile=2\]". A command is also provided for the termination of jobs that do not come online.

The fifth host options, named "my\_opts5", specifies 2 processes (with the same architecture as the master) on a Grid farm requiring the jobs to be launched using the project named "bnormal" and does not specify an upper limit on the number of cores to use, hence the default will apply. A command is also provided for the termination of jobs that do not come online. Note: The -b y option is required when accessing grid.

Call the **set\_host\_options** command to specify the host options.

```
pt_shell> set_host_options -name my_opts1
1
pt_shell> set_host_options -name my_opts2 -num_processes 2 \
ptopt018
1
pt_shell> set_host_options -name my_opts3 -num_processes 4 \
 -submit_command "/usr/bin/rsh -n" ptopt010
1
pt_shell> set_host_options -name my_opts4 -num_processes 5 \
 -submit_command "/lsf/bin/bsub -n 2 -R rusage\[mem=500\] -
R span\[ptile=2\]" \
 -terminate_command "/lsf/bin/bkill"
1
pt_shell> set_host_options -name my_opts5 -num_processes 2 \
```

```
-submit_command "/grid/bin/qsub -b y -P bnormal" \
-terminate_command "/grid/bin/qdel"
```

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
1] Launched : ...
...
Status : Forking successful
Stdout : Process group is (31811)
.Stderr : **<<EMPTY>>**

2] Launched : ...
...
Status : Forking successful
Stdout : Process group is (31846)
.Stderr : **<<EMPTY>>**

3] Launched : ...
...
Status : Forking successful
Stdout : Process group is (31891)
.Stderr : **<<EMPTY>>**

4] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17082)
.Stderr : **<<EMPTY>>**

5] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17153)
.Stderr : **<<EMPTY>>**

6] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17231)
.Stderr : **<<EMPTY>>**

7] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17309)
.Stderr : **<<EMPTY>>**

8] Launched : ...
...
```

```

Status : Forking successful
Stdout : Job <321813> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

9] Launched : ...
...
Status : Forking successful
Stdout : Job <321814> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

10] Launched : ...
...
Status : Forking successful
Stdout : Job <321815> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

11] Launched : ...
...
Status : Forking successful
Stdout : Job <321816> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

12] Launched : ...
...
Status : Forking successful
Stdout : Job <321817> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

13] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188655 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**

14] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188668 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**
```

---

```

Distributed farm creation timeout : 21600 seconds
Waiting for 14 (of 14) distributed hosts (Tue Nov 24 11:09:49 2009)
Waiting for 9 (of 14) distributed hosts (Tue Nov 24 11:09:59 2009)
Waiting for 7 (of 14) distributed hosts (Tue Nov 24 11:10:09 2009)
Waiting for 6 (of 14) distributed hosts (Tue Nov 24 11:10:19 2009)
Waiting for 4 (of 14) distributed hosts (Tue Nov 24 11:10:29 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:39 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:49 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:59 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:09 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:19 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:29 2009)
Waiting for 0 (of 14) distributed hosts (Tue Nov 24 11:11:39 2009)
```

---

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: D-2009.12
Date : Tue Nov 24 11:11:39 2009

```

| Options<br>Name | Host          | 32Bit | Num<br>Processes |
|-----------------|---------------|-------|------------------|
| my_opts1        | **localhost** | N     | 1                |
| my_opts2        | **ptopt018**  | N     | 2                |
| my_opts3        | ptopt010      | N     | 4                |
| my_opts4        | >>farm<<      | N     | 5                |
| my_opts5        | >>farm<<      | N     | 2                |

| Options<br>Name | #  | Host<br>Name | Job<br>ID | Process<br>ID | Status |
|-----------------|----|--------------|-----------|---------------|--------|
| my_opts1        | 1  | ptopt018     | 31811     | 31811         | ONLINE |
| my_opts2        | 2  | ptopt018     | 31846     | 31846         | ONLINE |
|                 | 3  | ptopt018     | 31891     | 31891         | ONLINE |
| my_opts3        | 4  | ptopt010     | 17082     | 17082         | ONLINE |
|                 | 5  | ptopt010     | 17153     | 17153         | ONLINE |
|                 | 6  | ptopt010     | 17231     | 17231         | ONLINE |
|                 | 7  | ptopt010     | 17309     | 17309         | ONLINE |
| my_opts4        | 8  | mddg131      | 321813    | 8373          | ONLINE |
|                 | 9  | mdyak036     | 321814    | 31490         | ONLINE |
|                 | 10 | mdyak726     | 321815    | 21995         | ONLINE |
|                 | 11 | mdcul037     | 321816    | 16787         | ONLINE |
|                 | 12 | mdyak040     | 321817    | 18278         | ONLINE |
| my_opts5        | 13 | peopf105     | 1188655   | 18524         | ONLINE |
|                 | 14 | peopf33      | 1188668   | 8798          | ONLINE |

Hosts limits:

| Name                | Cores limits<br>User |
|---------------------|----------------------|
| ** local process ** | none                 |
| my_opts1            | 1                    |
| my_opts2            | 2                    |
| my_opts3            | 3                    |
| my_opts4            | 2                    |
| my_opts5            | 1                    |

1

## Remote Connection Considerations =====

1. In order for the master process to interact with remote processes there is a connection between the master and the remote host. Each connection consumes 2 file descriptors from the operating system. Generally there is a limit on the number of file descriptors a user process may have simultaneously open. Typically a user process is allowed open 1024 simultaneous file descriptors. Considering the master PrimeTime process uses several descriptors for normal operations the maximum number of remote processes that can be launched is capped at 500 remote processes. All hosts added beyond 500 will be rejected by the start\_hosts command. The system user limit on the number of file descriptors allowed can be determined by calling the following command or equivalent in a UNIX terminal.

"limit descriptors"

2. When using 'rsh' to launch remote processes, depending on the host configuration there may be limits imposed on the user in relation to the number of allowed connections.

- The host being connected to may define a limit on the number of simultaneous connections a process may make to the host in the '/etc/xinetd.conf' file. The 'instances' keyword defines the maximum number of connections allowed. Once the number of connections goes beyond this, the circuit will fail to setup and a protocol failure will be returned as an error. If this occurs then the 'Stderr' line of a remote process launch will contain the following

"poll: protocol failure in circuit setup"

Action: The system limit on the number of incoming connections cannot be changed by a user (only a super-user). The only action that can be taken is to reduce the number of remote processes to be launched on the specific remote host so that the number is below the instance limit configured by that remote host.

- The host launching the remote processes may define a limit on the range of privileged ports accessible to the user. The range is typically 512-1024. Once all the privileged ports are exhausted, subsequent rsh connections will fail to setup and an out of sockets error will be returned as an error. If this occurs then the 'Stderr' line for a remote process launch will contain the following

"rcmd: socket: All ports in use"

Action: The system limit on the number of privileged ports available cannot be changed by a user (only a super-user). The only action that can be taken is to use a different protocol to launch subsequent processes beyond the number of privileged ports. e.g. ssh which does not use privileged ports. e.g. submit the remote process to a managed resource such as LSF, Grid etc.

## SEE ALSO

stop\_hosts(2)  
set\_host\_options(2)  
remove\_host\_options(2)  
report\_host\_usage(2)

## **start\_profile**

Start profiling PrimeTime commands.

### **SYNTAX**

```
Boolean start_profile
[-output directory_name]
[-verbose]
[-for_synopsys]
[-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
 When this option is specified, all commands will be tracked 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.

-for_synopsys
 Specifying this option will switch on the generation of code-level profiling and stopwatch data. This will produce a number of binary files in the profiling output directory. Send these files to Synopsys for recommendations. Normal profiling output will not be generated.

-include report_list
 Limit the profiling actions to selected items. Allowable list entries are: stopwatch, tcl_profile and perf_cpu.
```

### **DESCRIPTION**

The **start\_profile** command is used to initiate 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. Hence, 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 will also be 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 will be written to an output directory at the exit of a PrimeTime session or when the command **stop\_profile** is executed. Stopping of profiling will also create 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. Please note that profiling is performed on a PrimeTime process, not on a scenario. So if there are less hosts than scenarios, the profiling output will only appear in scenario output directories for scenarios which were first to execute on slave hosts.

## EXAMPLES

The following gives an example of profiling, where the reports are being written to the directory **reports/run\_profile**.

```
pt_shell> start_profile -output reports/run_profile
pt_shell> # Usual PT script
pt_shell> stop_profile
```

The example below shows how to enable Tcl profiling in the distributed multi-scenario analysis (DMSA). The profiling is first enabled on the DMSA master and then on DMSA slaves.

```
pt_shell> set multi_scenario_working_directory ./ms_data
pt_shell> set master_profile_dir $multi_scenario_working_directory/master_profile
pt_shell> start_profile -out $master_profile_dir
pt_shell> # Setting up DMSA
pt_shell> current_session -all
pt_shell> remote_execute {
? set outdir $multi_scenario_working_directory/[current_scenario]/slave_profile
? start_profile -out $outdir
?
}
pt_shell>
```

## SEE ALSO

**stop\_profile(2)**  
**current\_scenario(2)**

## **stop\_gui**

Stops the Primetime GUI.

### **SYNTAX**

string **stop\_gui**

### **ARGUMENTS**

None.

### **DESCRIPTION**

This command stops the Primetime GUI and returns to the pt\_shell prompt. It is ignored within scripts and at the pt\_shell prompt if the Primetime GUI has not been started.

### **EXAMPLES**

The following example stops the Primetime GUI and returns to the pt\_shell prompt.

```
Primetime> stop_gui
```

### **SEE ALSO**

[start\\_gui\(2\)](#)

## **stop\_hosts**

Stops all hosts that have been started.

### **SYNTAX**

```
int stop_hosts
```

### **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 ptopt021 using a 64-bit binary and specifies five different sets of host options.

The first host options, named "my\_opts1", specifies 1 process with the same architecture as the master on the same host as the master and does not specify an upper limit on the number of cores to use, hence the default will apply.

The second host options, named "my\_opts2", specifies 2 processes on the same host as the master but explicitly mentions the name of the master's host and specifies to use a maximum of 2 cores per process.

The third host options, named "my\_opts3", specifies 4 processes (with the same architecture as the master) on the host named ptopt010 using "rsh" to connect to ptopt010 and specifying to use a maximum of 3 cores per process.

The fourth host options named "my\_opts4", specifies 5 processes (with the same architecture as the master) on an LSF farm, requesting 500MB of memory and 2 slots per compute server and specifying to us a maximum of 2 cores per process. Notice that the farm job must be specified with the number of slots needed to support the number of cores to use hence the "-n 2 -R span\[ptile=2\]". A command is also provided for the termination of jobs that do not come online.

The fifth host options, named "my\_opts5", specifies 2 processes (with the same architecture as the master) on a Grid farm requiring the jobs to be launched using the project named "bnormal" and does not specify an upper limit on the number of cores to use, hence the default will apply. A command is also provided for the termination of jobs that do not come online. Note: The "-b y" option is required when accessing grid.

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
1
pt_shell> set_host_options -name my_opts2 \
 -num_processes 2 ptopt018
1
pt_shell> set_host_options -name my_opts3 -num_processes 4 \
 -submit_command "/usr/bin/rsh -n" ptopt010
1
pt_shell> set_host_options -name my_opts4 -num_processes 5 \
 -submit_command "/lsf/bin/bsub -n 2 -R rusage\[mem=500\] -"
R span\[ptile=2\]" \
 -terminate_command "/lsf/bin/bkill"
1
pt_shell> set_host_options -name my_opts5 -num_processes 2 \
 -submit_command "/grid/bin/qsub -b y -P bnormal" \
 -terminate_command "/grid/bin/qdel"
1

```

Call the **start\_hosts** command to start the slave processes.

```

pt_shell> start_hosts
1] Launched : ...
...
 Status : Forking successful
 Stdout : Process group is (31811)
 Stderr : **<<EMPTY>>**

2] Launched : ...
...
 Status : Forking successful
 Stdout : Process group is (31846)
 Stderr : **<<EMPTY>>**

3] Launched : ...
...
 Status : Forking successful
 Stdout : Process group is (31891)
 Stderr : **<<EMPTY>>**

4] Launched : ...
...
 Status : Forking successful
 Stdout : Process group is (17082)
 Stderr : **<<EMPTY>>**

5] Launched : ...
...
 Status : Forking successful
 Stdout : Process group is (17153)
 Stderr : **<<EMPTY>>**
```

**stop\_hosts**

1438

```
6] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17231)
.Stderr : **<<EMPTY>>**

7] Launched : ...
...
Status : Forking successful
Stdout : Process group is (17309)
.Stderr : **<<EMPTY>>**

8] Launched : ...
...
Status : Forking successful
Stdout : Job <321813> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

9] Launched : ...
...
Status : Forking successful
Stdout : Job <321814> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

10] Launched : ...
...
Status : Forking successful
Stdout : Job <321815> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

11] Launched : ...
...
Status : Forking successful
Stdout : Job <321816> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

12] Launched : ...
...
Status : Forking successful
Stdout : Job <321817> is submitted to default queue <normal>.
.Stderr : **<<EMPTY>>**

13] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188655 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**

14] Launched : ...
...
Status : Forking successful
Stdout : Your job 1188668 ("pt_shell") has been submitted
.Stderr : **<<EMPTY>>**
```

```


Distributed farm creation timeout : 21600 seconds
Waiting for 14 (of 14) distributed hosts (Tue Nov 24 11:09:49 2009)
Waiting for 9 (of 14) distributed hosts (Tue Nov 24 11:09:59 2009)
Waiting for 7 (of 14) distributed hosts (Tue Nov 24 11:10:09 2009)
Waiting for 6 (of 14) distributed hosts (Tue Nov 24 11:10:19 2009)
Waiting for 4 (of 14) distributed hosts (Tue Nov 24 11:10:29 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:39 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:49 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:10:59 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:09 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:19 2009)
Waiting for 2 (of 14) distributed hosts (Tue Nov 24 11:11:29 2009)
Waiting for 0 (of 14) distributed hosts (Tue Nov 24 11:11:39 2009)


```

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: D-2009.12
Date : Tue Nov 24 11:11:39 2009

```

| Options Name | Host          | 32Bit | Num Processes |
|--------------|---------------|-------|---------------|
| my_opts1     | **localhost** | N     | 1             |
| my_opts2     | **ptopt018**  | N     | 2             |
| my_opts3     | ptopt010      | N     | 4             |
| my_opts4     | >>farm<<      | N     | 5             |
| my_opts5     | >>farm<<      | N     | 2             |

| Options Name | # Host Name | Job ID | Process ID | Status |
|--------------|-------------|--------|------------|--------|
| my_opts1     | 1 ptopt018  | 31811  | 31811      | ONLINE |
| my_opts2     | 2 ptopt018  | 31846  | 31846      | ONLINE |
|              | 3 ptopt018  | 31891  | 31891      | ONLINE |
| my_opts3     | 4 ptopt010  | 17082  | 17082      | ONLINE |
|              | 5 ptopt010  | 17153  | 17153      | ONLINE |
|              | 6 ptopt010  | 17231  | 17231      | ONLINE |
|              | 7 ptopt010  | 17309  | 17309      | ONLINE |
| my_opts4     | 8 mddg131   | 321813 | 8373       | ONLINE |

stop\_hosts  
1440

|          |    |          |         |       |        |
|----------|----|----------|---------|-------|--------|
|          | 9  | mdyak036 | 321814  | 31490 | ONLINE |
|          | 10 | mdyak726 | 321815  | 21995 | ONLINE |
|          | 11 | mdcwp037 | 321816  | 16787 | ONLINE |
|          | 12 | mdyak040 | 321817  | 18278 | ONLINE |
| my_opts5 | 13 | peopf105 | 1188655 | 18524 | ONLINE |
|          | 14 | peopf33  | 1188668 | 8798  | ONLINE |

Hosts limits:

| Name                | Cores limits |
|---------------------|--------------|
|                     | User         |
| ** local process ** | none         |
| my_opts1            | 1            |
| my_opts2            | 2            |
| my_opts3            | 3            |
| my_opts4            | 2            |
| my_opts5            | 1            |

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
Shutdown Process 7
Shutdown Process 8
Shutdown Process 9
Shutdown Process 10
Shutdown Process 14
Shutdown Process 13
Shutdown Process 12
Shutdown Process 11
```

1

Call the **report\_host\_usage** command after stopping the hosts to report the host options created by the **set\_host\_options** command, as well as the real time data associated with the processes. Notice that the job and process ID are now -1 and the status is SHUTDOWN.

```
pt_shell> report_host_usage
```

```

Report : host_usage
```

Version: D-2009.12

Date : Tue Nov 24 11:11:39 2009

\*\*\*\*\*

| Options Name | Host          | 32Bit  | Num Processes |          |
|--------------|---------------|--------|---------------|----------|
| my_opts1     | **localhost** | N      | 1             |          |
| my_opts2     | **ptopt018**  | N      | 2             |          |
| my_opts3     | ptopt010      | N      | 4             |          |
| my_opts4     | >>farm<<      | N      | 5             |          |
| my_opts5     | >>farm<<      | N      | 2             |          |
| Options Name | # Host Name   | Job ID | Process ID    | Status   |
| my_opts1     | 1             | -1     | -1            | SHUTDOWN |
| my_opts2     | 2             | -1     | -1            | SHUTDOWN |
|              | 3             | -1     | -1            | SHUTDOWN |
| my_opts3     | 4             | -1     | -1            | SHUTDOWN |
|              | 5             | -1     | -1            | SHUTDOWN |
|              | 6             | -1     | -1            | SHUTDOWN |
|              | 7             | -1     | -1            | SHUTDOWN |
| my_opts4     | 8             | -1     | -1            | SHUTDOWN |
|              | 9             | -1     | -1            | SHUTDOWN |
|              | 10            | -1     | -1            | SHUTDOWN |
|              | 11            | -1     | -1            | SHUTDOWN |
|              | 12            | -1     | -1            | SHUTDOWN |
| my_opts5     | 13            | -1     | -1            | SHUTDOWN |
|              | 14            | -1     | -1            | SHUTDOWN |

Hosts limits:

| Name                | Cores limits |
|---------------------|--------------|
| ** local process ** | User         |
| my_opts1            | none         |
| my_opts2            | 1            |
| my_opts3            | 2            |
| my_opts4            | 3            |
| my_opts5            | 2            |

1

## SEE ALSO

start\_hosts(2)  
set\_host\_options(2)  
remove\_host\_options(2)  
report\_host\_usage(2)

stop\_hosts

1442

## **stop\_profile**

Stop profiling PrimeTime commands.

### **SYNTAX**

boolean **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**

```
int swap_cell
cell_list
swap_in
[-dont_preserve_constraints]
[-file file_name]
[-format db | verilog | vhdl]
```

### **"Data Types**

|                  |        |
|------------------|--------|
| <i>cell_list</i> | list   |
| <i>swap_in</i>   | string |
| <i>file_name</i> | 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.

## 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**

|                          |      |
|--------------------------|------|
| <i>from_list</i>         | list |
| <i>rise_from_list</i>    | list |
| <i>fall_from_list</i>    | list |
| <i>through_list</i>      | list |
| <i>rise_through_list</i> | list |
| <i>fall_through_list</i> | list |
| <i>to_list</i>           | list |
| <i>rise_to_list</i>      | list |
| <i>fall_to_list</i>      | list |
| <i>reason_list</i>       | 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 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**

```
int unset_rtl_to_gate_name
name
```

### **ARGUMENTS**

name

Indicates 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\_noise**

Performs static crosstalk noise analysis for the current design.

### **SYNTAX**

```
int update_noise
[-full]
```

### **ARGUMENTS**

**-full**

By default, the **update\_noise** command performs the noise analysis only if the design is not up to date for noise analysis. Using the **-full** option forces the **update\_noise** command to perform the noise analysis regardless whether the design is out of date or not.

### **DESCRIPTION**

Updates the crosstalk noise for the current design. The crosstalk noise is also automatically updated by commands or attributes that need the information, such as the **report\_timing** command.

The **set\_noise\_parameters** command controls the settings that are used during the noise analysis update.

The noise analysis is performed at each stage of the design, by calculating the worst case noise bump that aggressors induce on the victim line when the victim line is not switching. Also, if the **-enable\_propagation** option is used with the **set\_noise\_parameters** command, during the noise analysis, calculated noise can also propagate to the next stage and combine with the bumps induced by the aggressors of the next stage.

The induced bumps from the aggressors are calculated based on the characteristics of the driver of the victim net. If the **set\_steady\_state\_resistance** command is specified on the library cell of the driver cell, that information would be used for the calculation of induced bumps from the aggressors. Otherwise, the steady state current-voltage (I-V) characteristics of the library cell of the driver is used. If the I-V information does not exist in the library, the steady state resistance from the library is used. If none of these are available, the steady state resistance of the victim driver is estimated from the delay and slew tables.

By default, the analysis of beyond high and low rails are disabled. If the **-include\_beyond\_rails** option is used with the **set\_noise\_parameters** command, noise analysis is also performed for beyond high and low rail during the noise update.

### **EXAMPLES**

The following example updates the noise information for the current design.

```
pt_shell> update_noise
```

## SEE ALSO

```
report_noise(2)
set_noise_parameters(2)
set_steady_state_resistance(2)
si_noise_effort_threshold_within_rails(3)
si_noise_effort_threshold_within_rails(3)
```

## **update\_power**

Updates power information on the current design.

### **SYNTAX**

```
int update_power
```

### **DESCRIPTION**

Updates power for the current design. Power is also automatically updated by command that retrieves power results, such as command **report\_power** and most power attributes. Command **update\_power** explicitly prepares the design for further analysis. Note that you must set variable **power\_enable\_analysis** to TRUE before any power command.

Starting from 2008.12 release, **update\_power** can also perform power waveform generation and peak power calculation.

Variable **power\_analysis\_mode** can be used to specify the power analysis mode to perform. PrimeTime PX can perform different types of power analysis based on the mode setting. The default power analysis mode is "averaged". For more information, see the **power\_analysis\_mode** man page.

Command **set\_power\_analysis\_options** can be used to specify the options for power analysis. It must be set before **update\_power** to take effects in power analysis. Issuing command **set\_power\_analysis\_options** with no option can reset all the power analysis options to default. Use command **report\_power\_analysis\_options** to get the current analysis option settings. For more information, see the **set\_power\_analysis\_options** man page.

PrimeTime PX can consume either time-based (e.g. VCD file) or statistical switching activity information (e.g. SAIF file) for power calculation. For a particular analysis mode, appropriate activity information must be provided. For example, in time\_based power analysis mode, a VCD file must be provided. In averaged power analysis mode, any form of switching activity information is accepted. You can specify a VCD file by using command **read\_vcd**. The statistical switching activity can be specified by using command **read\_saif** or **set\_switching\_activity**.

For power estimation using vector free power analysis, all primary inputs and black boxes' pins are assigned with default switching activities. You can use **set\_case\_analysis** or **set\_switching\_activity** to improve accuracy. For example, using **set\_case\_analysis** to set a scan enable pin to 0 gives more accurate average power for normal operating mode. Setting reasonable switching activity values for set/reset pins by using the **set\_switching\_activity** command is also preferred.

In averaged power analysis mode, only averaged power results will be calculated.

In time\_based mode, power waveforms can be generated for the design, specific hierarchies or cells. Power waveforms are series of event-based power data calculated over time while processing VCD activity data. Meanwhile, peak power, which is the largest power value in the waveform, is also captured. The power waveforms are written into waveform files, which can later be displayed by waveform

viewing tools. The peak power is saved and can be reported by **report\_power**. Besides generating power waveforms, the averaged power results are also calculated.

If a gate-level zero\_delay VCD (indicated by **read\_vcd -zero\_delay**) or a RTL VCD (indicated by **read\_vcd -rtl**) is used in time\_based mode, cycle accurate peak power analysis will be performed. In cycle accurate peak power analysis, the default sampling interval is the clock cycle associated with the fastest clock in the design. Within the sampling interval the power waveform is a constant which represents the total power dissipated in that cycle. Cycle accurate peak power analysis provides cycle level accurate power behavior for the design. It can report maximum cycle power and the simulation time of the cycle in which the maximum power happens. The results are shown as the peak power reported by **report\_power** command and peak power attributes. The **-cycle\_accurate\_clock** and **-cycle\_accurate\_cycle\_count** options of the **set\_power\_analysis\_options** command can be used to specify the cycle for this analysis.

In leakage-variation power analysis mode, leakage variation analysis will be performed. For more information, see the **power\_enable\_leakage\_variation\_analysis** man page.

PrimeTime PX supports UPF domain-based power report feature for multivoltage designs. You can use **set\_current\_power\_domain** or **set\_current\_power\_net** commands to set the domains or supply nets of interest. Only the power dissipated on the specified domains or supply nets are calculated and reported. By default (if not specified), all the domains are included in power analysis. Use **get\_current\_power\_domain** or **get\_current\_power\_net** to show the current settings.

## EXAMPLES

This example performs time-based power analysis for time windows (0-20 and 50-end) :

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -time {0 20 50 -1} dump.vcd
pt_shell> update_power
```

This example performs averaged power analysis:

```
pt_shell> set power_analysis_mode averaged
pt_shell> read_saif -instance top -input file.saif
pt_shell> update_power
```

This example performs time\_based power analysis for instance u1.u2 from 100 ns to 500 ns. It also generates the results.out file with time-based power results in .out format.

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -time {100 500} dump.vcd
pt_shell> set_power_analysis_options -waveform_output results -waveform_format out -
cells {u1.u2}
pt_shell> update_power
```

The following example performs cycle accurate peak power analysis to generate a cycle accurate waveform by using an RTL VCD file as input:

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd myRTL.vcd -rtl
pt_shell> set_power_analysis_options -waveform_output capp_waveform -
waveform_format out
pt_shell> update_power
```

## SEE ALSO

```
current_power_domain(2)
current_power_net(2)
get_switching_activity(2)
read_saif(2)
read_vcd(2)
report_power(2)
report_power_analysis_options(2)
report_switching_activity(2)
set_case_analysis(2)
set_power_analysis_options(2)
set_switching_activity(2)
write_saif(2)
power_analysis_mode(3)
power_enable_analysis(3)
power_enable_leakage_variation_analysis(3)
```

## **update\_scope\_data**

Updates the scope data captured and stored in the scope file.

### **SYNTAX**

```
int update_scope_data
[-remove_blocks block_names]
[-remove_scenarios scenario_names]
[-merge_with other_scope_files]
[-keep_last]
file_name
```

### **Data Types**

|                          |        |
|--------------------------|--------|
| <i>block_names</i>       | list   |
| <i>scenario_names</i>    | string |
| <i>other_scope_files</i> | list   |
| <i>file_name</i>         | string |

### **ARGUMENTS**

**-remove\_blocks *block\_names***

Specifies the names of the blocks for which the scope data is to be removed from the given scope data file. If no scenario names are specified, by default, scope data for the block in all scenarios are removed.

**-remove\_scenarios *scenario\_names***

Specifies the names of the scenarios for which the scope data is to be removed from the given scope data file. If no block names are specified, by default, scope data for the specified scenarios in all blocks are removed.

**-merge\_with *other\_scope\_files***

Specifies all the other scope files from which to load the scope data and merge into the scope file being updated. The scope file being updated should not be put in this list.

**-keep\_last**

Specifies the precedence of the data when resolving duplicated scope data during merging. By default, when this option is not specified, the first scope data found takes precedence and is kept in the final data after merge is done. If you want the last loaded scope data to take precedence, specify this option to override the default behavior. Note that the *scope\_file* being updated is always considered the first. All the other scope files specified in the *-merge\_with* option are treated in the order they are specified and loaded. Note the *-keep\_last* option should only be used together with the *-merge\_with* option.

***file\_name***

Specifies the scope file that should be loaded and updated.

## DESCRIPTION

This command is used to update the data stored in the binary scope files. It provides users with 2 types of mutually exclusive operations to manage the scope data created during block-level analysis: (1) to remove certain scope data that are no longer needed, (2) to merge the scope data stored in other scope files together into one file.

Note that only the file currently being updated is changed after a successful updating operation. No changes are made to the contents of any other files. Also note that the changes to scope file being updated is generally not reversible.

## EXAMPLES

This example merge 2 scope files into file 'myBlock.scope'.

```
pt_shell> update_scope_data ./myBlock.scope -merge_with {./1.scope ./2.scope}
```

The following command removes the scope data captured for block 'Block\_A' at scenarios "XYZ" and "ABC".

```
pt_shell> update_scope_data myBlock.scope -remove_blocks Block_A \
 -remove_scenarios {XYZ ABC}
```

## SEE ALSO

```
create_ilm(2)
extract_model(2)
create_scenario(2)
check_block_scope(2)
hier_scope_check_defaults(3)
```

## **update\_timing**

Updates timing information on the current design.

### **SYNTAX**

```
string update_timing
[-full]
```

### **ARGUMENTS**

**-full**

Indicates that the entire timing analysis is to be performed from the beginning. The default is to perform an incremental analysis, which updates only out-of-date information and runs more quickly.

### **DESCRIPTION**

Updates timing for the current design. Timing is also automatically updated by commands that need the information, such as the **report\_timing** command. This command explicitly prepares the design for further analysis.

By default, the **update\_timing** command uses an efficient timing analysis algorithm that requires minimal computation effort and updates existing timing analysis information only where needed. You can override this default behavior using the **-full** option, which causes the entire timing update to be performed from the beginning. To avoid unnecessarily long run times, use the **-full** option only when you need to override incremental timing analysis. If the design is not timed, the **update\_timing** command has the same runtime independent of the **-full** option.

If the **si\_enable\_analysis** variable is set to **true** and a valid PrimeTime SI license is in place, the **update\_timing** command also performs crosstalk-aware timing analysis.

### **EXAMPLES**

The following example updates timing information for the current design.

```
pt_shell> update_timing
pt_shell> update_timing -full
```

### **SEE ALSO**

```
report_timing(2)
si_enable_analysis(3)
```

## **upf\_version**

Specify the version for the UPF file and syntax checking.

### **SYNTAX**

```
string upf_version
[version_id]
```

### **Data Types**

```
version_id string
```

### **ARGUMENTS**

```
version_id
 Specifies the UPF version for which the file or UPF commands are intended.
```

### **DESCRIPTION**

As the UPF standard matures, new updated versions of the UPF standard can occur. The **upf\_version** command can be used to specify the version to restrict syntax checking and semantics of UPF commands.

If called with an argument, returns the argument. If called with no argument, returns the current version number.

Currently only version 1.0 is supported. A warning is issued if you specify any other version.

### **EXAMPLES**

Here are sample outputs when you use the **upf\_version** command to query the current UPF version.

```
pt_shell> upf_version
```

Here is an example of the output that shows all UPF commands, such as commands that might be affected by a specific UPF standard.

```
pt_shell> help UPF
connect_supply_net # Connect supply net to supply ports and/or pins
create_power_domain # define power supply distribution network
create_power_switch # Define a switch in the power domain
create_supply_net # Create power or ground supply net object
create_supply_port # Create a port on power domain
get_power_domains # Create a collection of power domains
....
```

## **SEE ALSO**

`create_power_domain(2)`  
`report_power_domain(2)`  
`create_supply_net(2)`  
`create_supply_port(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**

```
int write_activity_waveforms
-interval cycle_size
[-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**

|                    |        |
|--------------------|--------|
| <i>cycle_size</i>  | float  |
| <i>level</i>       | int    |
| <i>signal_list</i> | list   |
| <i>module_list</i> | list   |
| <i>file_name</i>   | string |
| <i>time_list</i>   | list   |
| <i>filename</i>    | string |
| <i>window_size</i> | string |
| <i>path</i>        | 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:

$$\text{Average toggle rate} = \frac{\text{\# of toggles on all the signals for the interval}}{(\text{\# of signals}) * (\text{length of interval in nanoseconds})}$$

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:

$$\text{Coverage} = \frac{\text{\# of signals with at least one toggle}}{\text{\# of signals}} * 100\%$$

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**

|                |        |
|----------------|--------|
| <i>file</i>    | string |
| <i>pattern</i> | 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:

**write\_arrival\_annotations**

1474

```
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\_binary\_aocvm**

Creates a binary advanced on-chip variation (AOCV) file from an AOCV file.

### **SYNTAX**

```
int write_binary_aocvm
[-compress]
aocvm_file
binary_file
```

### **Data Types**

|                    |        |
|--------------------|--------|
| <i>aocvm_file</i>  | string |
| <i>binary_file</i> | string |

### **ARGUMENTS**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <i>-compress</i>   | Compresses the output binary AOCV file to reduce file size. |
| <i>aocvm_file</i>  | Specifies the name of the input AOCV file.                  |
| <i>binary_file</i> | Specifies the name of the output binary AOCV file.          |

### **DESCRIPTION**

The **write\_binary\_aocvm** command creates binary encoded AOCV files from ASCII AOCV files. This command is used to protect sensitive process related information.

The **read\_aocvm** command can read binary and compressed binary AOCV files created using the **write\_binary\_aocvm** command. No additional arguments are required to read binary or compressed binary AOCV files.

An advanced OCV derate table imported from a binary or compressed binary file is not visible in the **report\_aocvm** output.

### **EXAMPLES**

The following example shows how to create a binary advanced OCV file named "binary\_file" from an ASCII AOCV file named "aocvm\_file".

```
pt_shell> write_binary_aocvm aocvm_file binary_file
```

The following example shows how to create a compressed binary AOCV file named "small\_binary\_file" from an ASCII AOCV file named "aocvm\_file".

```
pt_shell> write_binary_aocvm -compress aocvm_file small_binary_file
```

## **SEE ALSO**

`read_aocvm(2)`  
`remove_aocvm(2)`  
`report_aocvm(2)`

## **write\_changes**

Outputs netlist changes that have been recorded during this session.

### **SYNTAX**

```
status write_changes
[-format ptsh | text | dctcl | icctcl | icc2tcl]
[-output file_name]
[-reset]
```

### **Data Types**

*file\_name*      string

### **ARGUMENTS**

**-format** ptsh | text | dctcl | icctcl | icc2tcl

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 and later releases
- **icc2tcl** - Tcl script for IC Compiler II

**-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 option 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

write\_changes

```
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.

Note:

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.

## **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
```

Note:

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 icctl
#####
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 -hier]
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_leakage(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 instances as a pt\_shell, dc\_shell, or dc\_shell-t script.

### **SYNTAX**

```
string write_context
[-timing]
[-environment]
[-design_rules]
[-constant_inputs]
[-derive_file_name]
[-prefix file_prefix]
[-extension file_extension]
[-script_directory directory]
[-separator name_separator]
[-output file_name]
[-format ptsh | dcsh | dctcl]
[-nosplit]
cell_list
```

### **Data Types**

|                       |        |
|-----------------------|--------|
| <i>file_prefix</i>    | string |
| <i>file_extension</i> | string |
| <i>directory</i>      | string |
| <i>name_separator</i> | string |
| <i>file_name</i>      | string |
| <i>cell_list</i>      | list   |

### **ARGUMENTS**

**-timing**

Writes only timing information; for example, clocks, input and output delays, and timing exceptions. By default, all context information is written.

**-environment**

Writes only environment-related information; for example, operating conditions (process, temperature, and voltage), wire load model, capacitive loads on input and output pins, and driving cell information on input pins. By default, all context information is written.

**-design\_rules**

Writes only design rules; for example, **max\_capacitance**, **max\_transition**, and **max\_fanout**. By default, all context information is written.

**-constant\_inputs**

Writes logic constants on input pins of the characterized instance. By default, all context information is written.

**-derive\_file\_name**

Derives the name of the script file from the instance name. This option and

the **-output** option are mutually exclusive. If neither option is specified, the script is written to standard output. If you use the **-prefix**, **-extension**, **-script\_directory**, or **-separator** option, you must also use the **-derive\_file\_name** option.

**-prefix** *file\_prefix*

If the **-derive\_file\_name** option is used, it specifies the prefix for the derived constraint file name. By default, no prefix is used. You cannot use the slash (/) or apostrophe (') characters in the prefix. This option and the **-output** option are mutually exclusive. If you use this option, you must also use the **-derive\_file\_name** option.

**-extension** *file\_extension*

If the **-derive\_file\_name** option is used, specifies the extension for the derived constraint file name; the default extension is **ptsh**. This option and the **-output** option are mutually exclusive. If you use this option, you must also use the **-derive\_file\_name** option.

**-script\_directory** *directory*

If the **-derive\_file\_name** option is used, specifies the directory in which the constraint file is to be generated; the default is the current directory. This option and the **-output** option are mutually exclusive. If you use this option, you must also use the **-derive\_file\_name** option.

**-separator** *name\_separator*

If the **-derive\_file\_name** option is used, specifies a single character string to use as a hierarchical separator in constructing the derived constraint file name generated for each instance; the default is the "at" sign (@). You cannot use the slash (/) or backslash (\) characters as name separators. This option and the **-output** option are mutually exclusive. If you use this option, you must also use the **-derive\_file\_name** option.

**-output** *file\_name*

Writes the script to *file\_name* file. This option and the **-derive\_file\_name** option are mutually exclusive; if neither is specified, the script is written to standard output. You also cannot use the **-prefix**, **-extension**, **-script\_directory**, or **-separator** option with the **-output** option.

**-format** *ptsh | dcsh | dctcl*

Specifies the output format for the script. The allowed values are **ptsh** (the default) for pt\_shell, **dcsh** for dc\_shell, and **dctcl** for dc\_shell-t.

**-nosplit**

The **-nosplit** option prevents line-splitting. This is most useful for doing diffs on previous scripts or for postprocessing the script.

**cell\_list**

Specifies a list of instances for which the context is to be written out as a script.

## DESCRIPTION

This command writes the timing context of the specified instances as a pt\_shell, dc\_shell, or dc\_shell-t script. The context must be captured using the

**characterize\_context** command. The script contains a series of pt\_shell, dc\_shell, or dc\_shell-t commands. When you execute these commands on the subdesign, PrimeTime initializes its timing environment to the information characterized from the context. This command is the mechanism used to export context information to other tools.

If the **write\_context** command is issued without options, the script is written for all context information, in pt\_shell format, to standard output.

Because the **characterize\_context** command does not capture delays and parasitics annotated on internal nets of the characterized instance, the generated script does not include this information. To export this information to module level tools, use the **write\_physical\_annotations** command. To augment the script generated by the **write\_context** command so that it imports the physical annotation files generated by the **write\_physical\_annotations** command, use the **-append** option of the **write\_physical\_annotations** command.

## EXAMPLES

The following example writes to standard output the timing-related context information for instances I1 and I2, in dc\_shell format.

```
pt_shell> write_context -timing -format dcsh {I1 I2}
```

The following command writes the environment and constant input context information for I2 to a pt\_shell script file named des1.

```
pt_shell> write_context -environment -constant_inputs -output des1 I2
```

The following command writes all context information in dc\_shell format. The two files FRST\_I1%X.scr and FRST\_I2.scr are generated in the directory ~/project/scripts.

```
pt_shell> write_context -derive_file_name -prefix FRST_
 -separator % -extension scr \
 -script_directory ~/projects/scripts \
 -format dcsh {I1/X I2}
```

The following command writes all context information in pt\_shell format to a file named I1.ptsh in the current directory.

```
pt_shell> write_context -derive_file_name I1
```

## SEE ALSO

**characterize\_context(2)**  
**remove\_context(2)**  
**report\_context(2)**  
**write\_physical\_annotations(2)**

## **write\_ilm\_netlist**

Writes out a flattened Verilog netlist corresponding to the interface logic for the current design.

### **SYNTAX**

```
int write_ilm_netlist
[-include_all_net_pins]
[-verbose]
file_name
```

### **Data Types**

file\_name                  string

### **ARGUMENTS**

**-include\_all\_net\_pins**

Indicates that all pins on nonclock interface logic nets and propagated clock interface logic nets are to be included in the netlist. By default, only pins that have the *is\_interface\_logic\_pin* attribute are written out in the Verilog netlist. Use this option if the interface logic model (ILM) is used in non-SDF flows and delay calculation is performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net. This option does not affect nonpropagated clock nets; that is, nets in the clock network that have user-specified source latency.

**-verbose**

Indicates that information about the number of cells and nets in the original design and in the ILM netlist is to be written to standard output.

file\_name

Specifies the name of the output file containing the flattened Verilog netlist information.

### **DESCRIPTION**

Writes out a flattened Verilog netlist corresponding to the interface logic on the current design. This represents the netlist of the interface logic model (ILM).

Use the **-include\_all\_net\_pins** option if the ILM is used in the non-SDF flows. You can use the **-verbose** option for approximating the reduction of the ILM size when compared with the original netlist.

For a discussion of ILM creation and the associated commands, see the **identify\_interface\_logic** man page.

## EXAMPLES

The following example writes out a Verilog netlist that includes statistics comparing the ILM with the original netlist.

```
pt_shell> write_ilm_netlist -verbose model.v
```

The following example writes out a Verilog netlist while keeping all pins on interface logic nets in the netlist.

```
pt_shell> write_ilm_netlist -include_all_net_pins model.v
```

## SEE ALSO

`get_ilm_objects(2)`  
`identify_interface_logic(2)`  
`write_ilm_parasitics(2)`  
`write_ilm_script(2)`  
`write_ilm_sdf(2)`

## **write\_ilm\_parasitics**

Writes out in Standard Parasitic Exchange Format (SPEF) or Synopsys Binary Parasitic Format (SBPF) all annotated parasitics on the interface logic for the current design.

### **SYNTAX**

```
int write_ilm_parasitics
[-input_port_nets]
[-constant_nets]
[-compress compression]
[-format SPEF | SBPF]
file_name
```

### **Data Types**

|                    |        |
|--------------------|--------|
| <i>compression</i> | string |
| <i>file_name</i>   | string |

### **ARGUMENTS**

|                              |                                                                                                                                                                         |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -input_port_nets             | Indicates that parasitic information is to be written out for nets connected to the input ports on the current design. By default, this information is not written out. |
| -constant_nets               | Indicates that parasitic information is to be written out for constant nets. By default, this information is not written out.                                           |
| -compress <i>compression</i> | Specifies that the file should be compressed. The only valid value for <i>compression</i> is <b>gzip</b> .                                                              |
| -format SPEF   SBPF          | Specifies the format of the file. The valid values are <b>SPEF</b> and <b>SBPF</b> . The default is <b>SPEF</b> .                                                       |
| <i>file_name</i>             | Specifies the name of the output SPEF file.                                                                                                                             |

### **DESCRIPTION**

This command writes out in SPEF format all annotated parasitics on the current design that are defined on its interface logic. This information is used as back-annotation information for the interface logic model (ILM).

For a discussion of ILM creation and the associated commands, see the **identify\_interface\_logic** man page.

## EXAMPLES

The following example writes to the SPEF file model.spef the parasitics that apply to interface logic.

```
pt_shell> write_ilm_parasitics model.spef
```

The following example includes parasitic information for the input ports and writes all parasitic information to the SPEF file model.spef.

```
pt_shell> write_ilm_parasitics -input_port_nets model.spef
```

## SEE ALSO

`get_ilm_objects(2)`  
`identify_interface_logic(2)`  
`write_ilm_netlist(2)`  
`write_ilm_script(2)`  
`write_ilm_sdf(2)`

## **write\_ilm\_script**

Writes constraints, assertions and exceptions for interface logic as a script for PrimeTime or Design Compiler.

### **SYNTAX**

```
int write_ilm_script
[-instance]
[-format ptsh | dcsch | dctcl]
[-compress compression]
[-nosplit]
file_name
```

### **Data Types**

|                    |        |
|--------------------|--------|
| <i>compression</i> | string |
| <i>file_name</i>   | string |

### **ARGUMENTS**

**-instance**  
Indicates that the script written out should be appropriate for use when the interface logic is instantiated at the chip level. That is, assertions and exceptions must refer to boundary pins on a block instead of referring to ports on a design. The script does not include environment information because that is defined by the chip-level design. Do not use this option to generate a script for verifying the standalone ILM against the original design.

**-format ptsh | dcsch | dctcl**  
Specifies the output format for the script. Allowed values are **ptsh** (the default) for pt\_shell, **dcsch** for dc\_shell, and **dctcl** for dc\_shell-t.

**-compress compression**  
Specifies that the script is to be compressed; by default, the script is written as standard text. The only valid value for the *compression* option is **gzip**.

**-nosplit**  
The **-nosplit** option prevents line-splitting. This is most useful for doing diffs on previous scripts, or for postprocessing the script.

**file\_name**  
Specifies the name of the script file to write. If the **-compress** option is also specified, the extension ".gz" is added if it is not already present.

### **DESCRIPTION**

Writes constraints, assertions and exceptions for interface logic as a script for pt\_shell, dc\_shell, or dc\_shell-t.

For a discussion of ILM creation and the associated commands, see the manual page for the **identify\_interface\_logic** command.

## EXAMPLES

The following example writes out in ptsh format the assertions and exceptions that apply to interface logic, to the file model.pt. This script would be used to verify an ILM against the original netlist.

```
pt_shell> write_ilm_script model.pt
```

The following example writes out the assertions and exceptions that can be applied to the ILM when it is instantiated at the chip level.

```
pt_shell> write_ilm_script -instance model.pt
```

## SEE ALSO

get\_ilm\_objects(2)  
identify\_interface\_logic(2)  
write\_ilm\_netlist(2)  
write\_ilm\_parasitics(2)  
write\_script(2)  
ilm\_ignore\_percentage(3)

## **write\_ilm\_sdf**

Writes out a Version 2.1-compliant Standard Delay Format (SDF) back-annotation file for the interface logic of the current design.

### **SYNTAX**

```
int write_ilm_sdf
[-input_port_nets]
[-output_port_nets]
[-significant_digits digits]
[-annotated]
[-no_edge]
[-compress compression]
[-version sdf_version]
file_name
```

### **"Data Types**

|                    |        |
|--------------------|--------|
| <i>digits</i>      | int    |
| <i>compression</i> | string |
| <i>sdf_version</i> | string |
| <i>file_name</i>   | string |

### **ARGUMENTS**

**-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.

**-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.

**-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.

**-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** commands.

**-no\_edge**

Indicates that the generated SDF is not to include any edges (posedge or negedge) for combinational IOPATHs.

**-compress *compression***

Specifies that the file should be compressed. The only valid value for *compression* is **gzip**.

```
-version sdf_version
 Selects which SDF version to use. Supported SDF versions are 1.0, 2.1, and
 3.0. SDF version 2.1 is the default.

file_name
 Specifies the name of the SDF file to write. If the -compress option is also
 specified, the extension ".gz" is added if it is not already present.
```

## DESCRIPTION

This command writes out in SDF format information on cell and net delays for the interface logic on the current design.

For information about ILM creation and the associated commands, see the **identify\_interface\_logic** man page.

## EXAMPLES

The following example writes out SDF information that applies to the interface logic; only SDF data that was previously back-annotated on a design is written out.

```
pt_shell> write_ilm_sdf -annotated model.sdf
```

The following example writes out an SDF file, model.sdf, for the interface logic. All delay values are included whether or not they had been back-annotated. Information is included for the input and output ports on the design. Four significant digits to the right of the decimal point are written in SDF delay triplets.

```
pt_shell> write_ilm_sdf -input_port_nets -output_port_nets -
significant_digits 4 model.sdf
in -0.25in
```

## SEE ALSO

```
get_ilm_objects(2)
identify_interface_logic(2)
write_ilm_netlist(2)
write_ilm_parasitics(2)
write_ilm_script(2)
write_sdf(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.

You normally use the **compare\_interface\_timing** command to compare two report files that have been generated by the **write\_interface\_timing** command.

The output report file contains the following sections:

**Slack Section** - Contains 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 Section** - Contains 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 Section** - Contains actual transition times on all ports for the four delay types: min\_fall, min\_rise, max\_fall, and max\_rise.

**Input Capacitance Section** - Contains actual total (lumped) or effective capacitances on boundary nets connected to each port.

**Design Rule Section** - Contains design rule information for all ports: max\_capacitance, min\_capacitance, max\_transition, max\_fanout, and fanout\_load.

**Noise Detection Section** - Contains noise slack information for all input ports for the four noise regions: below low, above low, below high, above high.

**Noise Calculation Section** - Contains driver steady-state resistance for all output ports for the four noise regions: below low, above low, below high, above high.

## 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 | SBPF
-no_name_mapping
-nets net_list
file_name
```

### **"Data Types**

|                  |        |
|------------------|--------|
| <i>net_list</i>  | list   |
| <i>file_name</i> | string |

### **ARGUMENTS**

```
-format SPEF | SBPF
 Specifies the format of the output parasitics file. Currently, the only
 allowed values are Standard Parasitic Exchange Format (SPEF) and Synopsys
 Binary Parasitics Format (SBPF).

-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 SBPF.

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 SBPF formats, and a *PrimeTime SI* license is required for SBPF.

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 SBPF 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 SBPF.

## EXAMPLES

The following example reads the top.spef parasitics file and then writes it out in SBPF 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 SBPF top_sbpf
Writing binary parasitics for 66 networks (66 RC, 0 PI, 0 LC)...
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**

|                        |        |
|------------------------|--------|
| <i>sdf_file</i>        | string |
| <i>sdf_version</i>     | string |
| <i>parasitics_file</i> | string |
| <i>script_file</i>     | string |
| <i>cell</i>            | 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 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 \
```

**write\_physical\_annotations**

1500

```
-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\_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]
```

### **Data Types**

|                  |        |
|------------------|--------|
| <i>file_name</i> | string |
| <i>cell_list</i> | list   |
| <i>value</i>     | 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, nonannotated SDPD information is also propagated and generated in the SAIF file. You can use the **-propagated** and **-exclude\_sdpd** options together to output the propagated simple switching activity and no SDPD information.

#### **-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.

## **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.

## **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 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**

|                      |        |
|----------------------|--------|
| <i>file_name</i>     | string |
| <i>compression</i>   | string |
| <i>category_list</i> | 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.

**-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**

|                        |        |
|------------------------|--------|
| <i>file_name</i>       | string |
| <i>sdc_version</i>     | string |
| <i>compression</i>     | string |
| <i>categories_list</i> | list   |

### **ARGUMENTS**

|                                 |                                                                                                                                                                                                                 |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>file_name</i>                | Specifies the name of the file to which the SDC script is to be written.                                                                                                                                        |
| -version <i>sdc_version</i>     | Specifies the version of SDC to write. Allowed values are <b>1.2</b> , <b>1.3</b> , <b>1.4</b> , <b>1.5</b> , <b>1.6</b> , <b>1.7</b> , <b>1.8</b> , <b>1.9</b> , <b>2.0</b> , and <b>latest</b> (the default). |
| -compress <i>compression</i>    | Specify that the script should be compressed. The only valid value for <i>compression</i> is <b>gzip</b> .                                                                                                      |
| -include <i>categories_list</i> | Write specified command categories only. The only valid value for <i>categories_list</i> is <b>exceptions</b> .                                                                                                 |
| -nosplit                        | The <b>-nosplit</b> 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**

```
string 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 sdf_context]
[-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 cells_list]
file_name
```

### **Data Types**

|                          |        |
|--------------------------|--------|
| <i>sdf_version</i>       | string |
| <i>digits</i>            | int    |
| <i>inst_name</i>         | string |
| <i>sdf_context</i>       | string |
| <i>sdf_map_file_list</i> | list   |
| <i>level</i>             | int    |
| <i>compression</i>       | string |
| <i>include_list</i>      | list   |
| <i>exclude_list</i>      | list   |
| <i>values_list</i>       | list   |
| <i>arc_type_list</i>     | list   |
| <i>file_name</i>         | 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** 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 ("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.

**file\_name**

Specifies the name of the SDF file to be written.

**-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. 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 cells_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.

## 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 technology 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 inter-block 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\_sdf\_constraints**

Writes to a disk file the constraints for the place and route layout tools.

### **SYNTAX**

```
int write_sdf_constraints
[-version ovi_version]
[-max_paths max_path_number]
[-nworst nworst_number]
[-slack_lesser_than slack_value]
[-cover_design]
[-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]
[-significant_digits digits]
[-compress compression]
[-context verilog | vhdl | none]
file_name
```

### **Data Types**

|                          |        |
|--------------------------|--------|
| <i>ovi_version</i>       | int    |
| <i>max_path_number</i>   | int    |
| <i>nworst_number</i>     | int    |
| <i>slack_value</i>       | float  |
| <i>from_list</i>         | list   |
| <i>rise_from_list</i>    | list   |
| <i>fall_from_list</i>    | list   |
| <i>through_list</i>      | list   |
| <i>rise_through_list</i> | list   |
| <i>fall_through_list</i> | list   |
| <i>to_list</i>           | list   |
| <i>digits</i>            | int    |
| <i>compression</i>       | string |
| <i>file_name</i>         | string |

### **ARGUMENTS**

```
-version ovi_version
 Specifies the SDF version in which timing information is to be written. The
 allowed values are 1.0 and 2.1 (default).

-max_paths max_path_number
 A positive, nonzero integer that specifies the maximum number of timing paths
 to select for each path group. The default is 1, meaning that only the path
 with the smallest timing slack is selected.
```

**-nworst** *nworst\_number*  
A nonzero integer that specifies the maximum number of paths to consider at each endpoint. The default is 1, meaning that only the worst path to an endpoint is considered. This option is ignored if you use the **-cover\_design** option.

**-slack\_lesser\_than** *slack\_value*  
A positive or negative floating point number that specifies the maximum slack value to use in selecting paths. Paths with slack values greater than *slack\_value* are not selected. Slack is calculated as the required arrival time minus the actual arrival time, so a negative slack value implies a constraint that has not been met.

**-cover\_design**  
Indicates to generate just enough path timing constraints to cover the worst path through every driver-load pin pair in the design. Path and net timing constraints are then based on this path set. The **-nworst**, **-to**, **-from**, and **-through** options are ignored when you use this option.

**-from** *from\_list*  
Specifies a list of pins or ports. Constraints are written to the constraints file for paths that start at the pins or ports on *from\_list*. This option is ignored if you use the **-cover\_design** option.

**-rise\_from** *rise\_from\_list*  
This option is similar to the **-from** option, except that the path must have a rising transition on the objects specified. This option is ignored if you use the **-cover\_design** option.

**-fall\_from** *fall\_from\_list*  
This option is similar to the **-from** option, except that the path must have a falling transition on the objects specified. This option is ignored if you use the **-cover\_design** option.

**-through** *through\_list*  
Specifies a list of path throughpoints (ports, pins, cells, or nets). Constraints are written to the constraints file for paths that go through points in the *through\_list*. This option is ignored if you use the **-cover\_design** option.

**-rise\_through** *rise\_through\_list*  
This option is similar to the **-through** option, except that the path must have a rising transition on the objects specified. This option is ignored if you use the **-cover\_design** option.

**-fall\_through** *fall\_through\_list*  
This option is similar to the **-through** option, except that the path must have a falling transition on the objects specified. This option is ignored if you use the **-cover\_design** option.

**-to** *to\_list*  
Specifies a list of pins or ports. Constraints are written to the constraints file for paths that end at the pins or ports on *to\_list*. This option is ignored if you use the **-cover\_design** option.

```

-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.

-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.

-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
 printed, the last pair of square bracket characters ("[" and "]") is not
 escaped. In the vhdl context, the last pair of parenthesis characters "("
 and ")") in a pin name is not escaped. In the default context none, all bus-
 delimiting characters are escaped with a backslash character ("always
 escaped.

file_name
 Specifies the name of the output file to which constraints for the current
 design are to be written.

```

## DESCRIPTION

This command writes specified constraints for the current design to a disk file. If you specify the **-cover\_design** option, just enough paths are generated so that the worst path through every driver-load pin pair is covered. The **-from**, **-rise\_from**, **-fall\_from**, **-to**, **-through**, **-rise\_through**, **-fall\_through**, and **-nworst** options are ignored when the **-cover\_design** option is used, and constraint generation is based on this "covering" path set. Using the **-max\_paths** option with **-cover\_design** is not recommended, because the objective of the **-cover\_design** option is to obtain a path set that covers the entire design.

To write constraints for the current design, arrival totals and slacks must be available throughout the design, not just at endpoints. To achieve this, you should set the **timing\_save\_pin\_arrival\_and\_slack** variable to **true** when this command is issued. If this variable has not been set to **true**, the 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, thus preventing the cost of an additional timing update.

## EXAMPLES

The following example writes constraints in SDF 2.1 format for the most critical path in each path group in the current design:

```
pt_shell> write_sdf_constraints cstr.sdf
```

The following example writes a constraints file in SDF 1.0 format for the 10 most critical paths in each path group in the current design "counter":

```
pt_shell> write_sdf_constraints -max_paths 10 -version 1.0 counter_cstr.sdf
```

The following example writes a constraints file in SDF 2.1 format with just enough paths so that the worst path through every driver-load pin pair is covered:

```
pt_shell> write_sdf_constraints -cover_design counter_cstr.sdf
```

## SEE ALSO

`write_sdf(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**

```
int 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]
paths_arcs_list
```

### **Data Types**

|                               |          |
|-------------------------------|----------|
| <i>type</i>                   | list     |
| <i>header_file_name</i>       | string   |
| <i>delay</i>                  | float    |
| <i>v1name</i>                 | string   |
| <i>v1</i>                     | float    |
| <i>v0name</i>                 | strong   |
| <i>v0</i>                     | float    |
| <i>trans</i>                  | float    |
| <i>file_name</i>              | string   |
| <i>spice_sub_circuit_file</i> | string   |
| <i>number_of_points</i>       | unsigned |
| <i>num</i>                    | float    |
| <i>number_of_samples</i>      | integer  |
| <i>precision</i>              | unsigned |
| <i>tran_size</i>              | float    |
| <i>tran_step</i>              | float    |

### **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 *vame*

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 *vame*

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 if the SPICE subcircuit has a different pin order from that of the .lib file.

**-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*.

**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  $(\text{sweep\_range}) / (\text{number\_of\_sweep\_points}-1)$  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]}\n{slew pin_or_port_name [meas_name]}\n{noise_peak pin_or_port_name [meas_name]}\n{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)`