cadence®

Genus Command Reference for Legacy UI

Product Version 19.1
November 2019

© 2015-2018 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

Alphabetical List of Commands 31

Preface 41

About This Manual 42
Additional References 42
Reporting Problems or Errors in Manuals 43
Customer Support 44
Cadence Online Support 44
Other Support Offerings 44
Supported User Interfaces 45
Messages 46
Man Pages 47
Command-Line Help 48
Getting the Syntax for a Command 48
Getting Attribute Help 48
Searching for Attributes 49
Searching For Commands When You Are Unsure of the Name 49
Documentation Conventions 50
Text Command Syntax 50

1
Navigation 51

basename 52
cd 53
dirname 55
dirs 56
filter 57
find 60
get_object_type 65
inout_mate 66

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

ls 67
popd 71
pushd 73
pwd 74
set_top_module 75
vdir_lsearch 76
vname 78
what_is 79
what_is_list 80

2
General 83

? 86
alias 87
apropos 88
attribute_exists 90
cdnshelp 91
check_ccr 92
clear 93
clear_redline_terminal 94
date 95
enable_transparent_latches 96
eval_common 97
exec_embedded_script 98
exit 100
get_attribute 101
get_liberty_attribute 103
get_read_files 104
help 105
include 106
is_attribute 107
is_common_ui_mode 108
lcd 109
license 110
license checkin 111

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license checkout 112
license feature 113
license list 114
license version 115
lls 116
lminus 117
lpopd 118
lpushd 119
lpwd 120
man 121
more 123
quit 125
redirect 126
reset_attribute 129
resume 131
search_doc 132
set_attribute 133
set_proc_verbose 135
sh 136
shell 137
source 138
stop_suspend 139
string_representation 140
suppress_messages 142
suspend 144
tcl_load 145
test_super_thread_servers 148
unsuppress_messages 149

3
Flowkit 151

check_flow 153
create_flow 155
create_flow_step 156
define_feature 158

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_flow 160
delete_flow_step 161
edit_flow 162
flowkit_template 163
get_feature 165
get_flow_config 166
is_flow 167
read_flow 168
report_flow 169
run_flow 170
schedule_flow 172
set_feature 173
set_flow_config 174
set_flowkit_read_db_args 175
set_flowkit_write_db_args 176
write_flow 177
write_flow_template 178

4
Metrics 179

create_metric_page 181
create_metric_table 182
create_metric_table_heading 184
create_snapshot 185
define_metric 187
delete_metric 189
delete_metric_page 190
delete_metric_run 191
delete_metric_table 192
delete_metric_table_heading 193
get_metric 194
get_metric_alias 196
get_metric_config 197
get_metric_definition 198
pop_snapshot_stack 199

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

push_snapshot_stack 200
read_metric 201
read_metric_config 202
report_metric 203
reset_metric_config 204
reset_metric_run_id 205
run_metric_category 206
set_metric 207
set_metric_alias 208
set_metric_header 210
update_metric 211
write_metric 213
write_metric_config 215

5
GUI Text 217

General GUI Text Commands 218
gui_balloon_info 219
gui_hide 219
gui_info 219
gui_legend 220
gui_raise 221
gui_reset 222
gui_resume 222
gui_selection 222
gui_show 223
gui_status 223
gui_suspend 224
gui_update 224
HDL Viewer GUI Text Commands 225
gui_hv_clear 226
gui_hv_get_file 226
gui_hv_load_file 226
gui_hv_set_indicators 227
Schematic Viewer GUI Text Commands 229

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_sv_clear . . . 230
gui_sv_cone . . . 230
gui_sv_get_instance . . . 231
gui_sv_grey . . . 231
gui_sv_highlight . . . 232
gui_sv_load . . . 233
gui_sv_snapshot . . . 234
gui_sv_toolbar_button . . . 234
Physical Viewer GUI Text Commands . . . 236
add_gui_shape . . . 238
deselect_obj . . . 240
gui_pv_airline_add . . . 241
gui_pv_airline_add_custom . . . 243
gui_pv_airline_delete . . . 244
gui_pv_airline_display . . . 245
gui_pv_airline_raw_add . . . 246
gui_pv_airline_raw_add_custom . . . 248
gui_pv_align_instance_to_boundary . . . 249
gui_pv_align_instances . . . 249
gui_pv_clear . . . 250
gui_pv_connectivity_airlines . . . 251
gui_pv_deselect . . . 252
gui_pv_display_collection . . . 253
gui_pv_draw_box . . . 254
gui_pv_draw_circle . . . 256
gui_pv_draw_line . . . 258
gui_pv_draw_net . . . 259
gui_pv_draw_triangle . . . 260
gui_pv_get_design . . . 261
gui_pv_grey . . . 262
gui_pv_highlight . . . 263
gui_pv_highlight_hier_instances . . . 265
gui_pv_highlight_update . . . 266
gui_pv_label . . . 267
gui_pv_lineup_instances . . . 268
gui_pv_new_viewer . . . 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_preferences 270
gui_pv_redraw 271
gui_pv_select 272
gui_pv_selection 273
gui_pv_snapshot 274
gui_pv_steiner 277
gui_pv_steiner_tree 278
gui_pv_toolbar_button 279
gui_pv_update 279
gui_pv_zoom_box 280
gui_pv_zoom_fit 280
gui_pv_zoom_in 281
gui_pv_zoom_out 281
gui_pv_zoom_to 283
select_obj 283

6
Chipware Developer 285

cwd 286
cwd check 287
cwd create_check 291
cwd report_check 294
hdl_create 297
hdl_create binding 299
hdl_create component 301
hdl_create implementation 303
hdl_create library 305
hdl_create operator 306
hdl_create package 307
hdl_create parameter 309
hdl_create pin 311

7
Input and Output 313

decrypt 317

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

encrypt . . . 318
export_critical_endpoints . . . 321
export_oa_db . . . 323
hdl_set_vlog_file_extension . . . 325
import_oa_db . . . 327
read_clip . . . 328
read_db . . . 329
read_def . . . 331
read_dfm . . . 332
read_dft_abstract_model . . . 334
read_encounter . . . 335
read_hdl . . . 336
read_ilm . . . 340
read_ilm_from_files . . . 341
read_io_speclist . . . 342
read_memory_view . . . 343
read_metric . . . 344
read_netlist . . . 345
read_physical . . . 347
read_pmbist_interface_files . . . 348
read_power_intent . . . 349
read_qrc . . . 350
read_saif . . . 351
read_sdc . . . 352
read_sdf . . . 355
read_sdp_file . . . 356
read_spef . . . 357
read_tcf . . . 358
read_vcd . . . 359
split_db . . . 360
write_atpg . . . 361
write_bsdl . . . 362
write_compression_macro . . . 363
write_db . . . 364
write_def . . . 366
write_design . . . 367

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_abstract_model . . . 369
write_dft_compression_test_points . . . 370
write_dft_deterministic_test_points . . . 371
write_dft_lbist_test_points . . . 372
write_do_ccd . . . 373
write_do_ccd cdc . . . 374
write_do_ccd compare_sdc . . . 375
write_do_ccd generate . . . 377
write_do_ccd propagate . . . 379
write_do_ccd validate . . . 381
write_do_clp . . . 382
write_do_lec . . . 385
write_do_verify . . . 390
write_do_verify cdc . . . 391
write_encounter . . . 393
write_et_atpg . . . 396
write_et_bsv . . . 397
write_et_lbist . . . 398
write_et_no_tp_file . . . 399
write_ett . . . 400
write_flow . . . 401
write_flow_template . . . 402
write_forward_saif . . . 403
write_hdl . . . 404
write_icgc_monitor . . . 408
write_io_speclist . . . 409
write_ldb . . . 410
write_logic_bist_macro . . . 411
write_metric . . . 412
write_name_mapping . . . 413
write_pmbist_interface_files . . . 414
write_pmbist_testbench . . . 415
write_power_intent . . . 416
write_saif . . . 417
write_scandef . . . 418
write_script . . . 419

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sdc 422
write_sdf 426
write_sdp_file 433
write_set_load 434
write_snapshot 435
write_spef 437
write_sv_wrapper 438
write_tcf 442
write_template 443
write_tempus 446

8
Constraints 449

clock_uncertainty 450
create_mode 453
define_clock 456
define_cost_group 461
derive_environment 462
external_delay 464
generate_constraints 468
get_interactive_constraint_modes 470
multi_cycle 471
path_adjust 475
path_delay 479
path_disable 482
path_group 485
phys_enable_ocv 488
phys_reset_ocv 489
propagate_constraints 490
set_design_mode 492
set_interactive_constraint_modes 493
specify_paths 494
validate_constraints 500

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

9
Elaboration and Synthesis . . . 503
delete_buffer_tree . . . 504
elaborate . . . 505
get_remove_assign_options . . . 509
merge_to_multibit_cells . . . 510
remap_to_dedicated_clock_library . . . 512
remove_assigns_without_optimization . . . 514
remove_clock_reconvergence . . . 517
remove_inserted_sync_enable_logic . . . 518
retime . . . 519
set_remove_assign_options . . . 522
syn_generic . . . 526
syn_map . . . 529
syn_opt . . . 531

10
Hierarchical Flow . . . 535
assemble_design . . . 536
generate_ilm . . . 538
read_ilm . . . 540
read_ilm_from_files . . . 541

11
Analysis and Report . . . 543
all . . . 547
all_connected . . . 548
all des . . . 549
all des inps . . . 550
all des insts . . . 551
all des outs . . . 552
all des seqs . . . 553
all lib . . . 555
all lib bufs . . . 556

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all lib ties 557
analyze_library_corners 558
check_design 560
check_timing_intent 565
clock_ports 567
compare_sdc 568
create_timing_bin 569
fanin 572
fanout 576
report 579
report area 584
report boundary_opto 587
report case_analysis 589
report cdn_loop_breaker 590
report cell_delay_calculation 592
report clock_gating 593
report clocks 600
report congestion 602
report datapath 603
report design_rules 608
report dft_chains 610
report dft_clock_domain_info 615
report dft_core_wrapper 617
report dft_registers 620
report dft_setup 624
report dft_violations 629
report disabled_transparent_latches 632
report gates 633
report hierarchy 637
report instance 639
report memory 642
report memory_cells 643
report messages 644
report min_pulse_width 647
report mode 648
report module 649

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report multibit_inferencing 650
report net_cap_calculation 657
report net_delay_calculation 658
report net_res_calculation 660
report nets 661
report opcg_equivalents 664
report ple 666
report port 668
report power 669
report power_intent 681
report power_intent_instances 685
report proto 688
report qor 690
report scan_compressibility 697
report sequential 699
report slew_calculation 702
report summary 703
report test_power 705
report timing 709
report units 721
report utilization 722
report yield 723
report_chains_longest_wires 724
report_chains_total_wirelength 726
report_chains_wires 728
report_cwd_check 730
report_dft_abstract_models 731
report_dft_trace_back 732
report_flow 734
report_logic_levels_histogram 735
report_metric 736
report_obj 737
report_scan_compression_logic 739
report_signoff_timing 741
statistics 742
statistics add_metric 744

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics log 746
statistics read 748
statistics remove_metric 749
statistics report 750
statistics reset 753
statistics run_stage_ids 754
statistics write 755
time_info 756
timestat 757
validate_timing 758

12
Physical 761

check_floorplan 764
check_placement 768
convert_polygon_to_boxes 770
create_clip 771
create_group 772
create_place_blockage 773
create_placement_blockage 775
create_placement_halo_blockage 777
create_region 778
create_route_blockage 780
create_routing_blockage 782
create_routing_halo_blockage 784
create_row 785
create_single_row 787
create_track 789
cut_power_domain_by_overlaps 790
def_move 791
delete_rows 792
duplicate_register 793
generate_ple_model 795
generate_reports 797
get_computed_shapes 799

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_obj_in_area 800
modify_power_domain_attr 801
move_blockage 803
move_instance 805
move_port 806
move_region 808
place_dft 809
place_dft_sequentials 810
predict_floorplan 811
read_clip 812
read_def 814
read_encounter 818
read_physical 819
read_sdp_file 821
read_spef 822
report congestion 823
report utilization 824
resize_blockage 825
resize_region 826
restore_congestion_map 827
specify_cell_pad 828
specify_floorplan 829
summary_table 831
update_congestion_map 833
update_group 834
update_instance_data 835
write_congestion_map 836
write_def 837
write_lib_lef 839
write_sdp_file 840
write_spef 841

13
Design for Test 843

add_analyzed_test_points 848

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_opcg_hold_mux 851
add_pmbist_access_method 852
analyze_scan_compressibility 854
analyze_testability 864
check_atpg_rules 867
check_dft_pad_configuration 869
check_dft_rules 870
check_dft_setup 877
compress_block_level_chains 878
compress_scan_chains 882
concat_scan_chains 897
configure_pad_dft 899
connect_cgic_test_pins_controlling_wrapper_cells 900
connect_compression_clocks 901
connect_dft_top_level 903
connect_opcg_segments 904
connect_scan_chains 906
create_chains_database 911
define_dft 912
define_dft abstract_segment 915
define_dft boundary_scan_segment 921
define_dft dft_configuration_mode 925
define_dft domain_macro_parameters 927
define_dft fixed_segment 929
define_dft floating_segment 931
define_dft jtag_instruction 933
define_dft jtag_instruction_register 937
define_dft jtag_macro 939
define_dft mbist_clock 944
define_dft opcg_domain 947
define_dft opcg_mode 951
define_dft opcg_trigger 953
define_dft osc_source 956
define_dft pmbist_direct_access 958
define_dft pmbist_port 962
define_dft preserved_segment 963

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft scan_chain 966
define_dft scan_clock_a 972
define_dft scan_clock_b 975
define_dft shift_enable 978
define_dft shift_register_segment 981
define_dft tap_port 983
define_dft test_bus_port 985
define_dft test_clock 990
define_dft test_mode 995
define_test_signal 1000
dft_trace_back 1008
fix_dft_violations 1010
fix_scan_path_inversions 1014
identify_domain_crossing_pins_for_cgic_and_scan_abstracts 1015
identify_multibit_cell_abstract_scan_segments 1016
identify_shared_wrapper_cells_in_design 1018
identify_shift_register_scan_segments 1019
identify_test_mode_registers 1021
insert_dft 1024
insert_dft boundary_scan 1026
insert_dft compression_logic 1031
insert_dft hard_repair 1042
insert_dft jtag_macro 1046
insert_dft lockup_element 1050
insert_dft logic_bist 1051
insert_dft opcg 1063
insert_dft pmbist 1065
insert_dft ptam 1071
insert_dft scan_power_gating 1074
insert_dft shadow_logic 1077
insert_dft shift_register_test_points 1082
insert_dft test_point 1083
insert_dft user_test_point 1089
insert_dft wir_signal_bits 1092
insert_dft wrapper_cell 1094
insert_dft wrapper_instruction_register 1102

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft wrapper_mode_decode_block 1104
insert_test_compression 1106
map_dft_unmapped_logic 1109
map_pmbist_cgc_to_cgic 1110
map_pmbist_ffn 1111
map_pmbist_ffsync 1112
read_dft_abstract_model 1113
read_io_speclist 1116
read_memory_view 1117
read_pmbist_interface_files 1120
replace_opcg_scan 1121
replace_scan 1123
report dft_chains 1125
report dft_clock_domain_info 1126
report dft_core_wrapper 1127
report dft_registers 1128
report dft_setup 1129
report dft_violations 1130
report opcg_equivalents 1131
report scan_compressibility 1132
report test_power 1133
report_chains_longest_wires 1134
report_chains_total_wirelength 1135
report_chains_wires 1136
report_scan_compression_logic 1137
reset_opcg_equivalent 1138
reset_scan_equivalent 1139
set_compatible_test_clocks 1140
set_opcg_equivalent 1142
set_scan_equivalent 1144
update_scan_chains 1146
write_atpg 1148
write_bsdl 1151
write_compression_macro 1154
write_dft_abstract_model 1162
write_dft_compression_test_points 1165

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_deterministic_test_points 1169
write_dft_lbist_test_points 1173
write_dft_opcg_domain_macro_rtl 1177
write_dft_opcg_trigger_macro_rtl 1178
write_dft_rtl_model 1179
write_et_atpg 1181
write_et_bsv 1188
write_et_lbist 1192
write_et_no_tp_file 1197
write_io_speclist 1199
write_logic_bist_macro 1201
write_memory_physical_map 1206
write_pmbist_interface_files 1207
write_pmbist_testbench 1210
write_scandef 1216

14
Low Power Synthesis 1219

clock_gating 1221
clock_gating add_xor_gating 1223
clock_gating connect_test 1225
clock_gating declone 1226
clock_gating import 1228
clock_gating insert_in_netlist 1230
clock_gating insert_obs 1231
clock_gating join 1233
clock_gating rebalance 1235
clock_gating remove 1236
clock_gating share 1238
clock_gating split 1240
read_saif 1242
read_tcf 1247
read_vcd 1252
report clock_gating 1256
report power 1257

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

state_retention 1258
state_retention connect_power_gating_pins 1259
state_retention swap 1260
write_forward_saif 1261
write_saif 1263
write_tcf 1265

15
Advanced Low Power Synthesis 1267

apply_power_intent 1268
check_cpf 1270
check_library 1272
check_power_domain 1278
check_power_intent 1279
commit_power_intent 1281
create_library_domain 1282
eval_1801 1283
optimize_iso_ls 1284
read_power_intent 1285
report low_power_cells 1287
report low_power_intent 1288
reset_power_intent 1289
verify_power_structure 1290
write_power_intent 1292

16
Design Manipulation 1295

bitblast_all_ports 1297
bitblast_ports 1298
change_link 1299
change_names 1301
clock_gating 1308
connect 1309
create_design 1310
create_hport_bus 1311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_inst 1312
create_primitive 1314
dedicate_module 1317
delete_unloaded_undriven 1318
disconnect 1319
edit_netlist 1320
edit_netlist bitblast_all_ports 1322
edit_netlist bitblast_port 1323
edit_netlist connect 1324
edit_netlist dedicate_subdesign 1326
edit_netlist delete 1328
edit_netlist disconnect 1329
edit_netlist flatten_complex_ports 1331
edit_netlist group 1332
edit_netlist hier_connect 1334
edit_netlist new_design 1335
edit_netlist new_instance 1336
edit_netlist new_port_bus 1338
edit_netlist new_primitive 1339
edit_netlist new_subport_bus 1341
edit_netlist ungroup 1342
edit_netlist uniquify 1343
flatten_complex_ports 1344
group 1345
insert_tiehilo_cells 1346
mv 1349
remove_cdn_loop_breaker 1351
reset_design 1353
rm 1355
ungroup 1357
uniquify 1359

17
Joules 1361

get_lib_domains 1362

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power 1363
report_sdb_annotation 1366
write_icgc_monitor 1368

18
Customization 1369

add_command_help 1370
define_attribute 1371
mesg_make 1376
mesg_send 1378
parse_options 1380

19
SDC Commands 1385

Introduction 1391
add_to_collection 1392
all_clocks 1395
all_connected 1396
all_fanin 1397
all_fanout 1399
all_inputs 1401
all_instances 1402
all_outputs 1403
all_registers 1404
append_to_collection 1406
compare_collections 1408
copy_collection 1409
create_clock 1410
create_generated_clock 1412
current_design 1416
current_design_name 1417
current_instance 1418
filter_collection 1419
foreach_in_collection 1421
get_arcs 1422

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_cells 1424
get_clocks 1428
get_designs 1430
get_generated_clocks 1432
get_leaf_cells 1435
get_leaf_nets 1436
get_leaf_pins 1437
get_libs 1438
get_lib_arcs 1440
get_lib_cells 1443
get_lib_pins 1445
get_nets 1448
get_object_name 1451
get_path_groups 1452
get_pins 1454
get_ports 1458
get_references 1461
getenv 1462
group_path 1463
index_collection 1467
list_property 1468
load_of 1469
remove_case_analysis 1470
remove_clock_gating_check 1471
remove_clock_latency 1472
remove_clock_transition 1473
remove_disable_clock_gating_check 1474
remove_disable_timing 1476
remove_from_collection 1477
remove_generated_clock 1479
remove_ideal_net 1480
remove_ideal_network 1481
remove_input_delay 1482
remove_output_delay 1483
report_property 1484
reset_clock 1485

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_gating_check 1486
reset_clock_groups 1487
reset_clock_latency 1488
reset_clock_sense 1489
reset_clock_transition 1490
reset_clock_uncertainty 1491
reset_data_check 1493
reset_disable_clock_gating_check 1495
reset_disable_timing 1496
reset_drive 1497
reset_driving_cell 1498
reset_generated_clock 1499
reset_ideal_network 1500
reset_input_delay 1501
reset_load 1503
reset_max_capacitance 1504
reset_max_fanout 1505
reset_max_time_borrow 1506
reset_max_transition 1507
reset_min_capacitance 1508
reset_min_fanout 1509
reset_min_pulse_width 1510
reset_min_transition 1511
reset_mode 1512
reset_output_delay 1513
reset_path 1515
reset_path_group 1517
reset_resistance 1518
reset_timing_derate 1519
reset_wire_load_mode 1521
reset_wire_load_model 1522
reset_wire_load_selection_group 1523
set_active_clocks 1524
set_annotated_check 1525
set_annotated_delay 1527
set_annotated_transition 1530

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_case_analysis 1531
set_clock_exclusive 1532
set_clock_gating_check 1533
set_clock_groups 1536
set_clock_latency 1539
set_clock_sense 1543
set_clock_skew 1545
set_clock_transition 1547
set_clock_uncertainty 1548
set_constraint_mode 1551
set_data_check 1552
set_disable_clock_gating_check 1556
set_disable_timing 1558
set_dont_touch 1560
set_dont_touch_network 1561
set_dont_use 1562
set_drive 1563
set_driving_cell 1566
set_equal 1570
set_false_path 1571
set_fanout_load 1577
set_hierarchy_separator 1578
set_ideal_latency 1579
set_ideal_net 1580
set_ideal_network 1581
set_input_delay 1582
set_input_transition 1586
set_lib_pins 1588
set_load 1590
set_load_unit 1592
set_logic_dc 1593
set_logic_one 1594
set_logic_zero 1595
set_max_capacitance 1596
set_max_delay 1598
set_max_dynamic_power 1603

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_fanout 1604
set_max_leakage_power 1605
set_max_time_borrow 1606
set_max_transition 1608
set_min_capacitance 1610
set_min_delay 1612
set_min_fanout 1615
set_min_pulse_width 1616
set_min_transition 1618
set_mode 1619
set_multicycle_path 1621
set_operating_conditions 1625
set_opposite 1627
set_output_delay 1628
set_path_adjust 1632
set_port_fanout_number 1635
set_size_only 1637
set_timing_derate 1638
set_time_unit 1641
set_unconnected 1642
set_units 1643
set_wire_load_mode 1645
set_wire_load_model 1646
set_wire_load_selection_group 1648
sizeof_collection 1650
sort_collection 1651

A
Applets 1653
Introduction 1654
applet 1655
applet avail 1656
applet install 1658
applet list 1659
applet load 1660

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet set_defaults 1662
applet update 1663
applet version 1664
applet whatis 1665

Index 1667

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Alphabetical List of Commands

Symbols

? 86

A

add_analyzed_test_points 848
add_command_help 1370
add_gui_shape 238
add_opcg_hold_mux 851
add_pmbist_access_method 852
add_to_collection 1392
alias 87
all des inps 550
all des insts 551
all des outs 552
all des seqs 553
all lib 555
all lib bufs 556
all lib ties 557
all_clocks 1395
all_connected 548, 1396
all_fanin 1397
all_fanout 1399
all_inputs 1401
all_outputs 1403
all_registers 1404
analyze_library_corners 558
analyze_scan_compressibility 854
analyze_testability 864
append_to_collection 1406
applet 1655
applet avail 1656
applet install 1658
applet list 1659
applet load 1660
applet set_defaults 1662
applet update 1663
applet version 1664
applet whatis 1665
apply_power_intent 1268
apropos 88
assemble_design 536
attribute_exists 90

B

basename 52
bitblast_all_ports 1297
bitblast_ports 1298

C

cd 53
change_link 1299
change_names 1301
check_atpg_rules 867
check_ccr 92
check_cpf 1270
check_design 560
check_dft_pad_configuration 869
check_dft_rules 870
check_dft_setup 877
check_floorplan 764
check_flow 153
check_library 1272
check_placement 768
check_power_domain 1278
check_power_intent 1279
check_timing_intent 565
clear 93
clear_redline_terminal 94
clock_gating 1221
clock_gating add_xor_gating 1223
clock_gating connect_test 1225
clock_gating declone 1226
clock_gating import 1228
clock_gating insert_in_netlist 1230
clock_gating insert_obs 1231
clock_gating join 1233
clock_gating rebalance 1235
clock_gating remove 1236
clock_gating share 1238
clock_gating split 1240
clock_ports 567
clock_uncertainty 450
commit_power_intent 1281
compare_collections 1408
compare_sdc 568
compress_block_level_chains 878

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

compress_scan_chains 882
concat_scan_chains 897
configure_pad_dft 899
connect 1309
connect_cgic_test_pins_controlling_wrapper_cells 900
connect_compression_clocks 901
connect_dft_top_level 903
connect_opcg_segments 904
connect_scan_chains 906
convert_polygon_to_boxes 770
copy_collection 1409
create_chains_database 911
create_clip 771
create_clock 1410
create_design 1310
create_flow 155
create_flow_step 156
create_generated_clock 1412
create_group 772
create_inst 1312
create_library_domain 1282
create_metric_table 182
create_metric_table_heading 184
create_mode 453
create_place_blockage 773
create_placement_blockage 775
create_placement_halo_blockage 777
create_port_bus 1313
create_primitive 1314
create_region 778
create_route_blockage 780
create_routing_blockage 782
create_row 785
create_single_row 787
create_snapshot 181, 185
create_timing_bin 569
create_track 789
current_design 1416
current_design_name 1417
current_instance 1418
cut_power_domain_by_overlaps 790
cwd 286
cwd check 287
cwd create_check 291
cwd report_check 294

D

date 95
decrypt 317
dedicate_module 1317
def_move 791
define_attribute 1371
define_clock 456
define_cost_group 461
define_dft 912
define_dft abstract_segment 915
define_dft boundary_scan_segment 921
define_dft dft_configuration_mode 925
define_dft domain_macro_parameters 927
define_dft fixed_segment 929
define_dft floating_segment 931
define_dft jtag_instruction 933
define_dft jtag_instruction_register 937
define_dft jtag_macro 939
define_dft mbist_clock 944
define_dft opcg_domain 947
define_dft opcg_trigger 953
define_dft osc_source 956
define_dft pmbist_direct_access 958
define_dft pmbist_port 962
define_dft preserved_segment 963
define_dft scan_chain 966
define_dft scan_clock_a 972
define_dft scan_clock_b 975
define_dft shift_enable 978
define_dft shift_register_segment 981
define_dft tap_port 983
define_dft test_bus_port 985
define_dft test_clock 990
define_dft test_mode 995
define_feature 158
define_metric 187
define_test_signal 1000
delete_buffer_tree 504
delete_flow 160
delete_flow_step 161
delete_metric 189
delete_metric_page 190
delete_metric_run 191
delete_metric_table 192
delete_metric_table_heading 193
delete_rows 792
delete_unloaded_undriven 1318
derive_environment 462
deselect_obj 240
dft_trace_back 1008
dirname 55
dirs 56
disconnect 1319

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

duplicate_register 793

E

edit_flow 162
edit_netlist 1320
edit_netlist bitblast_all_ports 1322
edit_netlist bitblast_port 1323
edit_netlist connect 1324
edit_netlist dedicate_subdesign 1326
edit_netlist disconnect 1329
edit_netlist flatten_complex_ports 1331
edit_netlist group 1332, 1345
edit_netlist hier_connect 1334
edit_netlist new_design 1335
edit_netlist new_instance 1336
edit_netlist new_port_bus 1338
edit_netlist new_primitive 1339
edit_netlist new_subport_bus 1341
edit_netlist ungroup 1342
edit_netlist uniquify 1343
elaborate 505
enable_transparent_latches 96
encrypt 318
eval_1801 1283
eval_common 97
exec_embedded_script 98
exit 100
export_critical_endpoints 321
export_oa_db 323
external_delay 464

F

fanin 572
fanout 576
filter 57
filter_collection 1419
find 60
fix_dft_violations 1010
fix_scan_path_inversions 1014
flatten_complex_ports 1344
flowkit_template 163
foreach_in_collection 1421

G

generate_constraints 468
generate_ilm 538
generate_ple_model 795
generate_reports 797
get_arcs 1422
get_attribute 101
get_cells 1424
get_clocks 1428
get_computed_shapes 799
get_designs 1430
get_feature 165
get_flow_config 166
get_generated_clocks 1432
get_interactive_constraint_modes 470
get_leaf_cells 1435
get_leaf_nets 1436
get_leaf_pins 1437
get_lib_arcs 1440
get_lib_cells 1443
get_lib_domains 1362
get_lib_pins 1445
get_liberty_attribute 103
get_libs 1438
get_metric 194
get_metric_alias 196
get_metric_config 197
get_metric_definition 198
get_nets 1448
get_obj_in_area 800
get_object_name 1451
get_object_type 65
get_path_groups 1452
get_pins 1454
get_ports 1458
get_read_files 104
get_references 1461
get_remove_assign_options 509
getenv 1462
group 1345
group_path 1463
gui_balloon_info 219
gui_hide 219
gui_hv_clear 226
gui_hv_get_file 226
gui_hv_load_file 226
gui_hv_set_indicators 227
gui_info 219
gui_legend 220
gui_pv_airline_add 241
gui_pv_airline_add_custom 243
gui_pv_airline_delete 244
gui_pv_airline_display 245

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_raw_add 246
gui_pv_airline_raw_add_custom 248
gui_pv_align_instance_to_boundary 249
gui_pv_align_instances 249
gui_pv_clear 250
gui_pv_connectivity_airlines 251
gui_pv_deselect 252
gui_pv_display_collection 253
gui_pv_draw_box 254
gui_pv_draw_circle 256, 260
gui_pv_draw_line 258
gui_pv_draw_net 259
gui_pv_get_design 261
gui_pv_grey 262
gui_pv_highlight 263
gui_pv_highlight_hier_instance 265
gui_pv_highlight_update 266
gui_pv_label 267
gui_pv_lineup_instances 268
gui_pv_new_viewer 268
gui_pv_preferences 270
gui_pv_redraw 271
gui_pv_select 272
gui_pv_selection 273
gui_pv_snapshot 274
gui_pv_steiner 277
gui_pv_steiner_tree 278
gui_pv_toolbar_button 279
gui_pv_update 279
gui_pv_zoom_box 280
gui_pv_zoom_fit 280
gui_pv_zoom_in 281
gui_pv_zoom_out 281
gui_pv_zoom_to 283
gui_raise 221
gui_reset 222
gui_resume 222
gui_selection 222
gui_show 223
gui_status 223
gui_suspend 224
gui_sv_clear 230
gui_sv_cone 230
gui_sv_get_instance 231
gui_sv_grey 231
gui_sv_highlight 232
gui_sv_load 233
gui_sv_snapshot 234
gui_sv_toolbar_button 234
gui_update 224

H

hdl_create 297
hdl_create binding 299
hdl_create component 301
hdl_create implementation 303
hdl_create library 305
hdl_create operator 306
hdl_create package 307
hdl_create parameter 309
hdl_create pin 311
hdl_set_vlog_file_extension 325
help 105

I

identify_domain_crossing_pins_for_cgic_and_scan_abstracts 1015
identify_multibit_cell_abstract_scan_segments 1016
identify_shared_wrapper_cells_in_design 1018
identify_shift_register_scan_segments 1019
identify_test_mode_registers 1021
import_oa_db 327
include 106
index_collection 1467
inout_mate 66
insert_dft 1024
insert_dft boundary_scan 1026
insert_dft compression_logic 1031
insert_dft hard_repair 1042
insert_dft jtag_macro 1046
insert_dft lockup_element 1050
insert_dft logic_bist 1051
insert_dft opcg 1063
insert_dft pmbist 1065
insert_dft ptam 1071
insert_dft scan_power_gating 1074
insert_dft shadow_logic 1077
insert_dft shift_register_test_points 1082
insert_dft test_point 1083
insert_dft user_test_point 1089
insert_dft wir_signal_bits 1092
insert_dft wrapper_cell 1094
insert_dft wrapper_instruction_register 1102
insert_dft

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

wrapper_mode_decode_block 1104
insert_test_compression 1106
insert_tiehilo_cells 1346
is_attribute 107
is_common_ui_mode 108
is_flow 167

L

lcd 109
license 110
license checkin 111
license checkout 112
license feature 113
license list 114
license version 115
list_property 1468
lls 116
lminus 117
load_of 1469
lpopd 118
lpushd 119
lpwd 120
ls 67

M

man 121
map_dft_unmapped_logic 1109
map_pmbist_cgc_to_cgic 1110
map_pmbist_ffn 1111
map_pmbist_ffsync 1112
merge_to_multibit_cells 510
mesg_make 1376
mesg_send 1378
modify_power_domain_attr 801
more 123
move_blockage 803
move_instance 805
move_port 806
move_region 808
multi_cycle 471
mv 1349

N

newlink reset_input_delay 1501

O

optimize_iso_ls 1284

P

parse_options 1380
path_adjust 475
path_delay 479
path_disable 482
path_group 485
phys_enable_ocv 488
phys_reset_ocv 489
place_dft 809
place_dft_sequentials 810
pop_snapshot_stack 199
popd 71
predict_floorplan 811
propagate_constraints 490
push_snapshot_stack 200
pushd 73
pwd 74

Q

quit 125

R

read_clip 812
read_db 329
read_def 814
read_dfm 332
read_dft_abstract_model 1113
read_encounter 818
read_flow 168
read_hdl 336
read_ilm 540
read_ilm_from_files 541
read_io_speclist 1116
read_memory_view 1117
read_metric 201
read_metric_config 202
read_netlist 345
read_physical 819
read_pmbist_interface_files 1120
read_power_intent 1285

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_qrc 350
read_saif 1242
read_sdc 352
read_sdf 355
read_sdp_file 821
read_spef 822
read_tcf 1247
read_vcd 1252
redirect 126
remap_to_dedicated_clock_library 512
remove_assigns_without_optimization 514
remove_case_analysis 1470
remove_cdn_loop_breaker 1351
remove_clock_gating_check 1471
remove_clock_latency 1472
remove_clock_reconvergence 517
remove_clock_transition 1473
remove_disable_clock_gating_check 1474
remove_disable_timing 1476
remove_from_collection 1477
remove_generated_clock 1479
remove_ideal_net 1480
remove_ideal_network 1481
remove_input_delay 1482
remove_inserted_sync_enable_logic 518
remove_output_delay 1483
replace_opcg_scan 1121
replace_scan 1123
report 579
report area 584
report boundary_opto 587
report case_analysis 589
report cdn_loop_breaker 590
report cell_delay_calculation 592
report clock_gating 593
report clocks 600
report congestion 602
report datapath 603
report design_rules 608
report dft_chains 610
report dft_clock_domain_info 615
report dft_core_wrapper 617
report dft_registers 620
report dft_setup 624
report dft_violations 629
report disabled_transparent_latches 632
report gates 633
report hierarchy 637
report instance 639
report memory 642
report memory_cells 643
report messages 644
report min_pulse_width 647
report module 649
report multibit_inferencing 650
report net_cap_calculation 657
report net_delay_calculation 658
report net_res_calculation 660
report nets 661
report opcg_equivalents 664
report ple 666
report port 668
report power 669, 1363
report power_intent 681
report power_intent_instances 685
report proto 688
report qor 690
report scan_compressibility 697
report sequential 699
report slew_calculation 702
report summary 703
report test_power 705
report timing 709
report units 721
report yield 723
report_chains_wires 728
report_cwd_check 730
report_dft_abstract_models 731
report_dft_trace_back 732
report_flow 734
report_logic_levels_histogram 735
report_metric 736
report_obj 737
report_property 1484
report_scan_compression_logic 739
report_sdb_annotation 1366
report_signoff_timing 741
reset_attribute 129
reset_clock 1485
reset_clock_gating_check 1486
reset_clock_groups 1487
reset_clock_latency 1488
reset_clock_sense 1489
reset_clock_transition 1490
reset_clock_uncertainty 1491
reset_data_check 1493
reset_design 1353
reset_disable_clock_gating_check 1495
reset_disable_timing 1496
reset_drive 1497

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_driving_cell 1498
reset_generated_clock 1499
reset_ideal_network 1500
reset_input_delay 1501
reset_load 1503
reset_max_capacitance 1504
reset_max_fanout 1505
reset_max_time_borrow 1506
reset_max_transition 1507
reset_metric_config 204
reset_metric_run_id 205
reset_min_capacitance 1508
reset_min_fanout 1509
reset_min_pulse_width 1510
reset_min_transition 1511
reset_mode 1512
reset_opcg_equivalent 1138
reset_output_delay 1513
reset_path 1515
reset_path_group 1517
reset_power_intent 1289
reset_resistance 1518
reset_scan_equivalent 1139
reset_timing_derate 1519
reset_wire_load_selection_group 1523
reset_wireload_mode 1521
reset_wireload_model 1522
resize_blockage 825
resize_region 826
restore_congestion_map 827
resume 131
retime 519
rm 1328, 1355
run_flow 170
run_metric_category 206

S

schedule_flow 172
search_doc 132
select_obj 283
set_active_clocks 1524
set_annotated_delay 1527
set_annotated_transition 1530
set_attribute 133
set_case_analysis 1531
set_clock_exclusive 1532
set_clock_gating_check 1533
set_clock_groups 1536
set_clock_latency 1539
set_clock_sense 1489, 1543
set_clock_skew 1545
set_clock_transition 1547
set_clock_uncertainty 1548
set_compatible_test_clocks 1140
set_constraint_mode 1551
set_data_check 1552
set_disable_clock_gating_check 1556
set_disable_timing 1558
set_dont_touch 1560
set_dont_touch_network 1561
set_dont_use 1562
set_drive 1563
set_driving_cell 1566
set_equal 1570
set_false_path 1571
set_fanout_load 1577
set_feature 173
set_flow_config 174
set_flowkit_write_db_args 175, 176
set_hierarchy_separator 1578
set_ideal_latency 1579
set_ideal_net 1580
set_ideal_network 1581
set_input_delay 1582
set_input_transition 1586
set_interactive_constraint_modes 493
set_lib_pins 1588
set_load 1590
set_load_unit 1592
set_logic_dc 1593
set_logic_one 1594
set_logic_zero 1595
set_max_capacitance 1596
set_max_delay 1598
set_max_dynamic_power 1603
set_max_fanout 1604
set_max_leakage_power 1605
set_max_time_borrow 1606
set_max_transition 1608
set_metric 207
set_metric_alias 208
set_metric_header 210
set_min_capacitance 1610
set_min_delay 1612
set_min_fanout 1615
set_min_pulse_width 1616
set_min_transition 1618
set_mode 1619
set_multicycle_path 1621
set_opcg_equivalent 1142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_operating_conditions 1625
set_opposite 1627
set_output_delay 1628
set_path_adjust 1632
set_port_fanout_number 1635
set_proc_verbose 135
set_remove_assign_options 522
set_resistance 1636
set_scan_equivalent 1144
set_size_only 1637
set_time_unit 1641
set_timing_derate 1638
set_top_module 75
set_unconnected 1642
set_units 1643
set_wire_load_selection_group 1648
set_wireload_mode 1645
set_wireload_model 1646
sh 136
shell 137
sizeof_collection 1650
sort_collection 1651
source 138
specify_cell_pad 828
specify_floorplan 829
specify_paths 494
split_db 360
state_retention 1258
state_retention connect_power_gating_pins 1259
state_retention swap 1260
statistics 742
statistics add_metric 744
statistics log 746
statistics read 748
statistics remove_metric 749
statistics report 750
statistics reset 753
statistics run_stage_ids 754
statistics write 755
stop_suspend 139
string_representation 140
summary_table 831
suppress_messages 142
suspend 144
syn_generic 526
syn_map 529
syn_opt 531

T

test_super_thread_servers 148
time_info 756
timestat 757

U

ungroup 1357
uniquify 1359
unsuppress_messages 149
update_congestion_map 833
update_group 834
update_instance_name 835
update_metric 211
update_scan_chains 1146

V

validate_constraints 500
validate_timing 758
vdir_lsearch 76
verify_power_structure 1290
vname 78

W

what_is 79
what_is_list 80
write_atpg 1148
write_bsdl 1151
write_compression_macro 1154
write_congestion_map 836
write_db 364
write_def 837
write_design 367
write_dft_abstract_model 1162
write_dft_compression_test_points 1165
write_dft_deterministic_test_points 1169
write_dft_lbist_test_points 1173
write_dft_opcg_domain_macro_rtl 1177
write_dft_opcg_trigger_macro_rtl 1178
write_dft_rtl_model 1179
write_do_ccd 373
write_do_ccd cdc 374
write_do_ccd compare_sdc 375
write_do_ccd generate 377

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd validate 381
write_do_clp 382
write_do_lec 385
write_do_verify 390
write_do_verify cdc 391
write_encounter 393
write_et_atpg 1181
write_et_bsv 1188
write_et_lbist 1192
write_et_no_tp_file 1197
write_ett 400
write_flow 177
write_flow_template 178
write_forward_saif 1261
write_hdl 404
write_icgc_monitor 1368
write_io_speclist 1199
write_ldb 410
write_lib_lef 839
write_logic_bist_macro 1201
write_memory_physical_map 1206
write_metric 213
write_metric_config 215
write_name_mapping 413
write_pmbist_interface_files 1207
write_pmbist_testbench 1210
write_power_intent 1292
write_saif 1263
write_scandef 1216
write_script 419
write_sdc 422
write_sdf 426
write_sdp_file 840
write_set_load 434
write_snapshot 435
write_spef 841
write_sv_wrapper 438
write_tcf 1265
write_template 443
write_tempus 446

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Preface

- About This Manual on page 42
- Additional References on page 42
- Reporting Problems or Errors in Manuals on page 43
- Customer Support on page 44
- Supported User Interfaces on page 45
- Messages on page 46
- Man Pages on page 47
- Command-Line Help on page 48
- Documentation Conventions on page 50

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

About This Manual

This manual provides a concise reference of the commands available to the user when using the Genus software with the legacy user interface.

This manual describes each command available within the Genus shell with its command options.

Additional References

The following sources are helpful references, but are not included with the product documentation:

- TclTutor, a computer aided instruction package for learning the Tcl language: http://www.msen.com/~clif/TclTutor.html.
- TCL Reference, *Tcl and the Tk Toolkit,* John K. Ousterhout, Addison-Wesley Publishing Company
- *Practical Programming in Tcl and Tk*, Brent Welch and Ken Jones
- IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language (IEEE Std.1364-1995)
- IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language (IEEE Std. 1364-2005)
- IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language (IEEE STD 1800-2009)
- IEEE Standard VHDL Language Reference Manual (IEEE Std. 1076-1987)
- IEEE Standard VHDL Language Reference Manual (IEEE Std. 1076-1993)
- IEEE Standard VHDL Language Reference Manual (IEEE Std. 1076-2008)

Note: For information on purchasing IEEE specifications go to http://shop.ieee.org/store/ and click on *Publications & Standards.*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Reporting Problems or Errors in Manuals

The Cadence® Help online documention, lets you view, search, and print Cadence product documentation. You can access Cadence Help by typing cdnshelp from your Cadence tools hierarchy.

Contact Cadence Customer Support to file a CCR if you find:

- An error in the manual
- An omission of information in a manual
- A problem using the Cadence Help documentation system

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Customer Support

Cadence offers live and online support, as well as customer education and training programs.

Cadence Online Support

The Cadence® online support website offers answers to your most common technical questions. It lets you search more than 40,000 FAQs, notifications, software updates, and technical solutions documents that give you step-by-step instructions on how to solve known problems. It also gives you product-specific e-mail notifications, software updates, case tracking, up-to-date release information, full site search capabilities, software update ordering, and much more. For more information on Cadence online support go to http://support.cadence.com

Other Support Offerings

- **Support centers**—Provide live customer support from Cadence experts who can answer many questions related to products and platforms.
- **Software downloads**—Provide you with the latest versions of Cadence products.
- **University software program support**—Provides you with the latest information to answer your technical questions.
- **Training Offerings—**Cadence offers the following training courses for Genus:
  - Genus Synthesis Solution
  - Basic Static Timing Analysis
  - Fundamentals of IEEE 1801 Low-Power Specification Format
  - Advanced Synthesis with Genus Synthesis Solution
  - Low-Power Synthesis Flow with Genus Synthesis Solution

  The courses listed above are available in North America. For further information on the training courses available in your region, visit Cadence Training or write to training_enroll@cadence.com.

  Note: The links in this section open in a new browser.
- **Video Library**

  Several videos are available on the support website: Genus: Video Library

For more information on the support offerings go to http://www.cadence.com/support

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Supported User Interfaces

Genus supports the following user interfaces:

- **Unified User Interface.** Genus, Innovus and Tempus offer a fully unified Tcl scripting language and GUI environment. This unified user interface (also referred to as **Stylus** common UI) streamlines flow development and improves productivity of multi-tool users.

  When you start Genus, you will by default start with the common UI. You will see the following prompt:

  ```
  @genus:root: 1>
  ```

- **Legacy User Interface.** Genus can also operate in legacy mode which supports RTL Compiler commands/attributes and scripting.

  To start Genus with legacy UI, you can

  - Start the tool with legacy UI as follows:

    ```
    %genus -legacy_ui -files script
    ....
    legacy_genus:/>
    ```

  - Switch to legacy UI if you started the tool with the default common UI.

    ```
    %genus
    @genus:root: 1> set_db common_ui false
    legacy_genus:/>
    ```

Important

This document provides information specific to the legacy user interface.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Messages

- You can get detailed information for each message issued in your current Genus run using the `report messages` command.

  ```
  legacy_genus:/> report messages
  ```

  The report also includes a summary of how many times each message was issued

- You can also get specific information about a message.

  For example, to get more information about the `TUI-613` message, you can type the following command:

  ```
  legacy_genus:/> vls -a TUI-613
  message:TUI/TUI-613 (message)
      Attributes:
        base_name = TUI-613
        count = 0
        escaped_name = TUI/TUI-613
        help = The user_speed_grade is only applicable to datapath subdesigns.
        id = 613
        name = TUI/TUI-613
        obj_type = message
        print_count = 0
        priority = 1
        screen_print_count = 0
        severity = Warning
        type = The attribute is not applicable to the object.
  ```

If you do not get the details that you need or do not understand a message, either contact Cadence Customer Support to file a CCR or email the message ID you would like improved to synthesis_pubs@cadence.com.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Man Pages

In addition to the Command and Attribute References, you can also access information about the commands and attributes using the man pages in Genus.

To use man pages from UNIX shell:

1. Set your environment to view the correct directory:

   ```
   setenv MANPATH $CDN_SYNTH_ROOT/share/synth/man_legacy
   ```

2. Access the manpage by either of the following ways:

   - Enter the name of the command or attribute that you want. For example:

     - `man check_dft_rules`
     - `man default_power_rail`

   - Specify a section number with `man` command to look for the command or attribute information in the specific section of the on-line manual.

     Commands are in section 1, attributes are in section 2, and messages are in section 3 of the on-line manual. In the absence of section number, `man` will search through sections 1, 2, 3 (in this sequence) and display the first matching manual page.

     This is useful in cases where both commands and attributes exist with the same name. For example:

     - `man 1 retime`

       will display manhelp for command named `retime`

     - `man 2 retime`

       will display manhelp for attribute named `retime`

   Note: Refer to man for more information on the `man` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Command-Line Help

You can get quick syntax help for commands and attributes at the Genus command-line prompt. There are also enhanced search capabilities so you can more easily search for the command or attribute that you need.

Note: The command syntax representation in the Genus documentation does not necessarily match the information that you get when you type `help` *`command_name`* in the tool. In many cases, the order of the arguments is different. Furthermore, the syntax in this document includes all of the dependencies, where the help information does this only to a certain degree.

If you have any suggestions for improving the command-line help, please e-mail them to synthesis_pubs@cadence.com

Getting the Syntax for a Command

Type the `help` command followed by the command name.

For example:

```
legacy_genus:/> help path_adjust
```

This returns the syntax for the `path_adjust` command.

Getting Attribute Help

Type the following:

```
legacy_genus:/> get_attribute -h attribute_name *
```

For example:

```
legacy_genus:/> get_attribute -h max_transition *
```

This returns the help for the `max_transition` attribute and shows on which object types the attribute can be specified.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Searching for Attributes

You can get a list of all the available attributes by typing the following command:

```
legacy_genus:/> get_attribute * * -help
```

You can type a sequence of letters after the `set_attribute` command and press `Tab` to get a list of all attributes that contain those letters.

```
legacy_genus:/> set_attr li<T
```

Returns the list of all attributes starting with `li`.

Searching For Commands When You Are Unsure of the Name

You can use help to find a command if you only know part of its name, even as little as one letter.

- You can type a single letter and press `Tab` to get a list of all commands that start with that letter.

  For example:

  ```
  legacy_genus:/> a<Tab>
  ```

  This returns the following commands:

  ```
  add_command_help                add_opcg_hold_mux
  add_to_collection               after
  alias                           all_clocks
  all_connected                   all_fanin
  all_fanout                      all_inputs
  all_instances                   all_outputs
  all_registers                   analyze_library_corners
  analyze_scan_compressibility    analyze_testability
  append                          append_to_collection
  applet                          apply
  apply_power_intent              apropos
  array                           assemble_design
  attribute_exists                auto_execok
  auto_import                     auto_load
  auto_load_index                 auto_qualify
  ```

- You can type a sequence of letters and press `Tab` to get a list of all commands that start with those letters.

  For example:

  ```
  legacy_genus:/> path_<Tab>
  ```

  This returns the following commands:

  ```
  path_adjust  path_delay  path_disable  path_group
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Documentation Conventions

To aid the readers understanding, a consistent formatting style has been used throughout this manual.

- UNIX commands are shown following the `unix>` string.
- Genus commands are shown following the `legacy_genus:/>` string.

Text Command Syntax

The list below defines the syntax conventions used for the Genus text interface commands.

`literal`	Nonitalic words indicate keywords you enter literally. These keywords represent command or option names.
`arguments and options`	Words in italics indicate user-defined arguments or information for which you must substitute a name or a value.
|	Vertical bars (OR-bars) separate possible choices for a single argument.
[ ]	Brackets indicate optional arguments. When used with OR-bars, they enclose a list of choices from which you can choose one.
{ }	Braces indicate that a choice is required from the list of arguments separated by OR-bars. Choose one from the list. `{ argument1 | argument2 | argument3 }`
`{ }`	Braces, used in Tcl commands, indicate that the braces must be typed in.
...	Three dots (...) indicate that you can repeat the previous argument. If the three dots are used with brackets (that is, `[argument]...)`, you can specify zero or more arguments. If the three dots are used without brackets (`argument...)`, you must specify at least one argument.
{ }	Braces in bold-face type must be entered literally.
#	The pound sign precedes comments in command files.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

1

Navigation

- basename on page 52
- cd on page 53
- dirname on page 55
- dirs on page 56
- filter on page 57
- find on page 60
- get_object_type on page 65
- inout_mate on page 66
- ls on page 67
- popd on page 71
- pushd on page 73
- pwd on page 74
- vdir_lsearch on page 76
- vname on page 78
- what_is on page 79
- what_is_list on page 80

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

basename

`basename` *`pathname`*

Removes the leading directory names of the virtual path name and returns only the object name. This command behaves similarly to the UNIX `basename` command.

Options and Arguments

`pathname`	Specifies the virtual path name of the object, including the object name.

Examples

- The following example removes the directory name of the `CW_absval` ChipWare component and only returns the component name:

```
legacy_genus:/> basename \
    /hdl_libraries/CW/components/CW_absval/comp_architectures/CW_absval
CW_absval
```

- The following example uses the `basename` command with the `dirname` command to return the name of the library to which the `ND2X1` library cell belongs:

```
legacy_genus:/> set libcell /libraries/LIB/libcells/ND2X1
legacy_genus:/> basename [dirname [dirname $libcell]]
```

Related Information

Related commands: dirname on page 55

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cd

`cd [ vdir ]`

Sets the current directory in the design hierarchy and navigates the design hierarchy. This command is similar to its UNIX counterpart, but used in the virtual directory. A description of the design hierarchy is given in the *Genus User Guide for Legacy UI*.

Options and Arguments

vdir	Specifies the name of the vdir to be set as the current vdir. The ".", "..", and "/"have the same meaning as their UNIX counterparts (current vdir, parent vdir, and root vdir respectively). You can use wildcards when they do not produce an ambiguous reference (more than one match).

Examples

- The following command returns you to the top of the design hierarchy:

```
legacy_genus:/designs> cd
legacy_genus:/> pwd
/
```

- The following command specifies the absolute path to the target vdir:

```
legacy_genus:/> cd /designs/alu/timing/exceptions
```

- The following command specifies the relative path to the target vdir:

```
legacy_genus:/> cd designs/alu
legacy_genus:/designs/alu> cd ./timing/exceptions
```

- The following command changes the current vdir to a parent vdir:

```
legacy_genus:/designs/alu/timing/exceptions> cd ../../subdesigns
legacy_genus:/designs/alu/subdesigns> pwd
/designs/alu/subdesigns
```

- The following command uses wildcards in the path specification:

```
legacy_genus:/designs/cmplx_alu/subdesigns/addinc> cd ../../tim*/exc*
legacy_genus:/designs/cmplx_alu/timing/exceptions> pwd
/designs/cmplx_alu/timing/exceptions
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:

dirs on page 56
ls on page 67
popd on page 71
pushd on page 73
pwd on page 74

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dirname

`dirname` *`pathname`* `[-times` *`integer`*`]`

Removes the object name of the specified path name and only returns the directory name. This command behaves similarly to the UNIX `dirname` command.

Options and Arguments

`pathname`	Specifies the path name of the object, including the object name.
`-times` *`integer`*	Specifies the number of times to apply dirname. *Default:* `1`

Examples

- The following example removes the `CW_absval` ChipWare component name and only returns its directory name:

```
legacy_genus:/>dirname \
    /hdl_libraries/CW/components/CW_absval/comp_architectures/CW_absval
/hdl_libraries/CW/components/CW_absval/comp_architectures
```

- The following command applies the `dirname` command 3 times to the specified path.

```
legacy_genus:/> dirname -times 3 \ /hdl_libraries/CW/components/CW_
absval/comp_architectures/CW_absval
/hdl_libraries/CW/components
```

- The following example uses the `basename` command with the `dirname` command to return the name of the library to which the `ND2X1` library cell belongs:

```
legacy_genus:/> set libcell /libraries/LIB/libcells/ND2X1
legacy_genus:/> basename [dirname [dirname $libcell]]
```

Related Information

Related commands: basename on page 52

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dirs

```
dirs
```

Displays the contents of the design virtual directory (vdir) stack. This command is similar to its UNIX counterpart and is used in conjunction with the `pushd` and `popd` commands.

Example

- The following commands respectively add designs and libraries to the design vdir stack, then display the contents of the vdir stack:

```
legacy_genus:/> pushd designs
/designs /
legacy_genus:/designs> pushd /libraries
/libraries /designs /
legacy_genus:/libraries> dirs
/libraries /designs /
```

- The following commands respectively remove the last added directories and then display the contents of the vdir stack:

```
legacy_genus:/libraries> popd
/designs /
legacy_genus:/designs> popd
/
legacy_genus:/> dirs
/
```

Related Information

Related commands:

ls on page 67
popd on page 71
pushd on page 73
pwd on page 74

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

filter

```
filter [-invert] [-vname]
     [-expr string] [-regexp]
     attribute_name attribute_value [object_list]
```

Filters a set of objects based on the values of the given attributes using the pattern matching mechanism from the `glob` Tcl command.

This command provides a powerful means of selecting objects within the design hierarchy at a more discrete level than is allowed by the directory structure alone.

Use the `get_attribute` command to list the attributes available for each object type.

Options and Arguments

attribute_name	Specifies the name of an attribute to use as filter. This argument is required. A compound string (containing spaces) should be represented as a list either by using double-quotes or braces ({ }).
attribute_value	Specifies the value of an attribute to use as filter.
-expr *string*	Specifies an expression that can be used to compare attribute values. Applies to numerical expressions only.
-invert	Filters out objects that match the expression and returns those that do not.
object_list	Specifies a Tcl list of objects to filter.
-regexp	Overrides the default Tcl glob pattern matching with Tcl regular expression matching.
-vname	Allows to specify the Verilog name instead of the Genus design hierarchy path name

Examples

- The following command finds the list of all matching library cells on which the `preserve` attribute has been set to true:

  ```
  legacy_genus:/> filter preserve true [find . -libcell *]
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command stores the Tcl list of all matching cells returned by the `filter` command in a variable for use in scripting later.

```
legacy_genus:/> set preserved_cells [filter preserve true [find . -libcell *]]
```

- The following command embeds the Tcl list of all matching cells returned by the filter command as part of a larger command.

```
legacy_genus:/> report timing -through [filter preserve true [find . -libcell *]]
```

- The following Tcl code fragment sets the variable `result` to all instances that start with the letter `g` and whose corresponding library cell starts with `inv`:

```
set result {}
foreach inst [find . -inst g*] {
  if {[string match "inv*" [get_att libcell $inst]]}
     {lappend result $inst}
}
puts $result
```

- The following example returns only returns those pins that have the `preserve` attribute set to either `true` or `false` and ignores those with `size_ok` values:

```
legacy_genus:/> filter -regexp preserve {true|false} [find / -pin *]
```

- The following command finds all the instances of cell `bufx1` in library `mylib` in design `dut_shell`:

```
filter -regexp libcell {.*bufx1} [find /designs/dut_shell -instance *]
```

- Use the `ls -dir` command to format the output:

```
legacy_genus:/> ls -dir [filter preserve true [find . -libcell *]]
/libraries/penny/libcells/ANTENNA/
/libraries/penny/libcells/FILL1/
/libraries/penny/libcells/RF1R1WX2/
/libraries/penny/libcells/RF2R1WX2/
```

- Use the ls -dir command and the redirect arrow to redirect the output to the specified file:

```
legacy_genus:/> ls -dir [filter preserve true [find . -libcell *]] > filter.txt
```

  You can also append arrows (">>").

- The following command returns the design whose tns value equals 0.

```
legacy_genus:/> get_attr tns [find / -design mydesign]
0
legacy_genus:/> filter -expr == tns 0 [find / -design *]
/designs/mydesign
```

- The following command returns the design whose tns value is larger than -1.

```
legacy_genus:/> filter -expr > tns -1 [find / -design *]
/designs/mydesign
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands: ls on page 67
get_attribute on page 101
set_attribute on page 133

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

find

```
find [root_path]
     [-maxdepth integer] [-mindepth integer]
     [-regexp expression] [-invert expression]
     [-ignorecase] [-split] [-vname] {-option...|-*} object
```

Searches the design hierarchy for the specified types of objects and returns a Tcl list containing the full paths to any matching objects. This list can then be used by other commands to operate on groups of objects. The `find` command supports the `*` and `?` wildcard characters.

Note: You cannot use the following options when doing an explicit find (for example, using `-instance name`): `-mindepth`, `-maxdepth` `-regexp`, `-invert`, `-ignorecase`, `-split`, `-vname`.

Tip

The `find` command is very powerful but overusing it can increase the execution time or lead to memory issues. To avoid issues, be as specific as possible when specifying the object to match. For example,

- Traverse the design recursively on hierarchical instances, instead of starting from the root directory
- Use the `-vname` option where possible.

Options and Arguments

`-ignorecase`	Ignores the case (upper case or lower case) of the parameters. Alternatively, specifies the search to be case insensitive.
`-invert` *`expression`*	Excludes the objects specified by the regular expression.
`-maxdepth` *`level`*	Descends no more than the specified number (non-negative integer) of levels below *`root_path`*. A level of `0` searches only the `root_path`. *Default*: infinity
`-mindepth` *`integer`*	Skips the specified number (non-negative integer) of levels below *`root_path`* before finding objects. A level of `1` searches all objects except *`root_path`*.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Default: `0`
`object`	Specifies the name of the object to match. The name can include wildcard characters.
`-option`	Specifies the type of object you want to find. You can specify multiple options or look in all types by specifying *. Check the command help for a list of the valid object types.
`-regexp` *`expression`*	Specifies to find the specified regular expression.
`root_path`	Specifies the name of the directory from where to start searching. The name can include wildcard characters. By default, an explicit search is done for the specified object. During an explicit search every object directory is searched, instead of starting from *`root_path`*. Specifying a *`root_path`* reduces the numbers of locations where the `find` command will search which reduces the execution time.
`-split`	Causes the command to return one object per line. This option is for interactive use only: the Tcl result is set to null and the objects are only printed to the screen.
`-vname`	Removes the container directories in the path name and returns Verilog style names where appropriate. This option will only work on the following objects: `pin`, `port`, `net`, `subdesign`, and `instance`.

Examples

- The following command searches for any object type whose name contains `add`, starting from the current directory (`/designs`):

```
legacy_genus:/designs> find . * *add*
/designs/alu/instances_hier/ops1_add_25 /designs/alu/subdesigns/addinc64
```

- The following command finds all registers in all designs:

```
legacy_genus:/> find des* -instance *seq/*
/designs/alu/instances_hier/RC_CG_HIER_INST_0/instances_seq/RC_CGIC_INST
/designs/alu/instances_seq/aluout_reg_7 /designs/alu/instances_seq/aluout_
reg_6 /designs/alu/instances_seq/aluout_reg_4 /designs/alu/instances_
seq/aluout_reg_0 /designs/alu/instances_seq/aluout_reg_3
/designs/alu/instances_seq/aluout_reg_5 /designs/alu/instances_seq/aluout_
reg_2 /designs/alu/instances_seq/aluout_reg_1 /designs/alu/instances_
seq/zero_reg
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command finds all input ports in design `alu`:

```
legacy_genus:/> find des*/alu -port ports_in/*
```

```
{/designs/alu/ports_in/opcode[2]} {/designs/alu/ports_in/opcode[1]}
{/designs/alu/ports_in/opcode[0]} {/designs/alu/ports_in/data[7]}
{/designs/alu/ports_in/data[6]} {/designs/alu/ports_in/data[5]}
{/designs/alu/ports_in/data[4]} {/designs/alu/ports_in/data[3]}
{/designs/alu/ports_in/data[2]} {/designs/alu/ports_in/data[1]}
{/designs/alu/ports_in/data[0]} {/designs/alu/ports_in/accum[7]}
{/designs/alu/ports_in/accum[6]} {/designs/alu/ports_in/accum[5]}
{/designs/alu/ports_in/accum[4]} {/designs/alu/ports_in/accum[3]}
{/designs/alu/ports_in/accum[2]} {/designs/alu/ports_in/accum[1]}
{/designs/alu/ports_in/accum[0]} /designs/alu/ports_in/clock
/designs/alu/ports_in/ena /designs/alu/ports_in/reset
```

- The following command searches for an external delay whose name starts with `in`, starting from the `designs` directory:

```
legacy_genus:/designs> find designs -external_delay in*
```

```
/designs/alu/timing/external_delays/in_del_1
```

- The following command finds all designs that are four characters:

```
legacy_genus:/> find . -designs ????
/designs/test
```

- The following command finds all design names with four characters that end with the letter "i":

```
legacy_genus:/> find . -designs ???i
/designs/topi
```

- The following command performs a case insensitive search for the design `TEST`:

```
legacy_genus:/> find . -ignorecase -design test
/designs/TEST
```

- To find hierarchical objects, you can just specify the top-level object instead of the root or current directory. Doing so can provide faster results because it minimizes the number of hierarchies that Genus traverses. In the following example, if we wanted to only find the output pins for `inst1`, the first specification is more efficient than the second. The second example not only traverses more hierarchies, it also returns `inst2` instances.

```
legacy_genus:/> find inst1 -pin out*
```

```
{/designs/woodward/instances_hier/inst1/pins_out/out1[3]}
```

```
legacy_genus:/>find / -pin out*
```

```
{/designs/woodward/instances_hier/inst1/pins_out/out1[3]}
{/designs/woodward/instances_hier/inst2/pins_out/out1[3]}
```

- The following example uses the `find` command to return a list of all the instances in a small design.

```
legacy_genus:/> find / -instance *
```

```
/designs/MOD69/instances_hier/inst1 /designs/MOD69/instances_
hier/inst1/instances_comb/g21
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The `-vname` option removes the container directories (in this case `instance_hier`) and presents the list more concisely:

```
legacy_genus:/> find / -instance -vname *
inst1
inst1/g21
```

This option is useful when you want to present the object in a report because the name is more concise. The disadvantage of the shortened name is that it may no longer refer to a unique object because an instance, pin, net, and subport may all share the same Verilog name.

- The following example uses the `-regexp` option to return all message objects that contain at least `VLOGPT-6`:

```
legacy_genus:/> find / -regexp (VLOGPT-6+) -messsages * *
```

- The following example uses the `-invert` option to return all library objects except for LIB2:

```
legacy_genus:/> find / -invert "LIB2" -library *
/libraries/LIB1 /libraries/LIB3
```

- The following example returns all combinational instances named `g58` or `g59` in the Verilog name style:

```
legacy_genus:/> find / -regexp {g[5][8-9]} -vname -instance instances_comb/*
```

- Use the `ls -dir` command to format the output row over row:

```
legacy_genus:/> ls -dir [find / -instance -vname *]
```

```
/designs/quea/instances_hier/inst1/
/designs/quea/instances_hier/inst1/instances_comb/g41/
/designs/quea/instances_hier/inst1/instances_comb/g42/
/designs/quea/instances_hier/inst1/instances_comb/g43/
/designs/quea/instances_hier/inst1/instances_comb/g44/
```

- Use the ls -dir command and the redirect arrow to redirect the output to the specified file:

```
legacy_genus:/> ls -dir [find / -instance -vname *] > quea.txt
```

- You can also append arrows (">>").

- The following examples show the difference between the result with and without `-split`.

```
legacy_genus:/designs> set rc [find / -inst a_reg_*]
```

```
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_0
/designs/DTMF_CHIP/instances_hier//DMA_INST/instances_seq/a_reg_1
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_2
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_3
```

```
legacy_genus:/designs> echo $rc
```

```
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_0
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_1
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_2
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/designs> set rc [find / -split -inst a_reg_*]
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_3
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_2
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_1
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_0
legacy_genus:/designs> echo $rc
legacy_genus:/designs>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_object_type

```
get_object_type
     object...
```

Returns a string describing the type of object passed to the command as its argument. The command will work only if a single object is specified.

This command is mostly used for writing Tcl scripts (rather than being an interactive command). It is useful for checking the types of arguments passed to the Tcl procedures.

Options and Arguments

object	Specifies the object for which you want to know the type.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

inout_mate

`inout_mate {subport_bus|port_bus|subport|port|pin|pgpin}`

Distinguishes bidirectional pins, ports, portables, supports, and subport_busses inout objects so you can find the input object and the corresponding output object.

Note: The command will return NULL for a pgpin.

These bidirectional inout objects are represented as two distinct objects: the input pin, port, port_bus, support, and subport_bus object and the corresponding output object.The pair cannot be broken apart, such as using an `edit_netlist rm` command to delete just one of the two objects, because together, they represent the inout object.

Examples

- The following example is a design called `top` that has a primary inout port named `io`, which Genus represents as two distinct ports:

  ```
  /designs/top/ports_in/io
  /designs/top/ports_out/io
  ```

  - Use the `inout_mate` command on the input object to find the corresponding output object:

    ```
    legacy_genus:/> inout_mate ports_in/io
    /designs/top/ports_out/io
    ```

  - Use the `inout_mate` command on the output object to find the corresponding input object

    ```
    legacy_genus:/> inout_mate ports_out/io
    /designs/top/ports_in/io
    ```

- The following example is an hierarchical instance called `s4` that has an inout port named `io`, which Genus represents as two distinct subports:

  ```
  /designs/top/instances_hier/s4/subports_in/io
  /designs/top/instances_hier/s4/subports_out/io
  ```

  - Use the `inout_mate` command on the input object to find the corresponding output object:

    ```
    legacy_genus:/> inout_mate s4/subports_in/io
    /designs/top/instances_hier/s4/subports_out/io
    ```

  - Use the `inout_mate` command on the output object to find the corresponding input object:

    ```
    legacy_genus:/> inout_mate s4/subports_out/io
    /designs/top/instances_hier/s4/subports_in/io
    ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

ls

```
ls [-computed] [-attribute] [-long] [-dir] [-R]
     [-regexp string] [-width integer] [object]... [>file]
```

Lists information about any objects in the design hierarchy (designs, library cells, clocks, and so on). This command is similar to its UNIX counterpart, but used in the virtual directory (vdir). The vdir contains all objects related to designs, libraries, flows, internal commands & messages, and more.

Options and Arguments

`-attribute`	List the attributes for the specified object whose values are different from the default values.
`-computed`	Lists all computed attributes. Computed attributes are potentially very time consuming to process and are therefore by default not listed.
`-dir`	Lists only the vdir name not its contents.
`file`	Specifies the name of the file to which to list the information.
`-long`	Lists the contents (long listing) of the vdir.
`object`	Specifies the vdir for which you want to list information. *Default*: current vdir
`-regexp` *`string`*	Specifies a regular expression to filter attribute names.
`-R`	Recursively lists the encountered subdirectories.
`-width` *`integer`*	Specifies the width of the screen that can be used to show the information. *Default*: `80`

Examples

- The following command lists the contents of the `libraries` vdir:

```
legacy_genus:/> ls libraries
/libraries:
./                em333s/           sm333s/
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command shows information of cell `buf1` in regular and long listing format:

```
legacy_genus:/libraries/tutorial/libcells/buf1> ls
./            A             Y

legacy_genus:/libraries/tutorial/libcells/buf1> ls -long
Total: 3 items
./                  (libcell)
A                   (libpin)
Y                   (libpin)
```

- The following command lists only the attributes of `buf1` whose values are different from the default values:

```
legacy_genus:/libraries/tutorial/libcells/buf1> ls -attribute
Total: 3 items
./                  (libcell)
    Attributes:
      area = 1
      buffer = true
      cell_leakage_power = 0.0 nW
      combinational = true
      liberty_attributes = area 1.0
A/                  (libpin)
    Attributes:
      capacitance = 25.0 25.0 femtofarads
      fanout_load = 1.000 fanout_load units
      input = true
      liberty_attributes = capacitance 0.025 direction input
      max_transition = 4500.0
      outgoing_timing_arcs = /libraries/tutorial/libcells/buf1/Y/inarcs/A_n90
Y/                  (libpin)
    Attributes:
      capacitance = 0.0 0.0 femtofarads
      fanout_load = 0.000 fanout_load units
      function = A
      incoming_timing_arcs = /libraries/tutorial/libcells/buf1/Y/inarcs/A_n90
      liberty_attributes = direction output function A
      max_transition = 4500.0
      output = true
```

- The following command lists all attributes (except for the computed attributes) for `buf1`, even those with the default value:

```
legacy_genus:/libraries/tutorial/libcells/buf1> ls -long -attribute
Total: 3 items

./                  (libcell)
    All attributes:
      adder = false
      area = 1.0
      async_clear =
      async_preset =
      avoid = false
      buffer = true
      cell_delay_multiplier = 1.0
      ...
      usable = true
      user_defined =
      width = no_value
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
A                   (libpin)
    All attributes:
      async_clear_phase = none
      async_preset_phase = none
      capacitance = 25.0 25.0 femtofarads
      clock_gate_clock_pin = false
      clock_gate_enable_pin = false
      clock_gate_obs_pin = false
      clock_gate_out_pin = false
      clock_gate_reset_pin = false
      clock_gate_test_pin = false
      ...
      tristate = false
      user_defined =
Y                   (libpin)
    All attributes:
      async_clear_phase = none
      async_preset_phase = none
      capacitance = 0.0 0.0 femtofarads
      clock_gate_clock_pin = false
      clock_gate_enable_pin = false
      clock_gate_obs_pin = false
      clock_gate_out_pin = false
      clock_gate_reset_pin = false
      clock_gate_test_pin = false
      ...
      tristate = false
      user_defined =
```

- The following command uses wildcard strings and the results of a `find` command:

```
legacy_genus:/libraries> ls -long [find . -libcell A*]
/libraries/cg/libcells/AND2A:
Total: 4 items
./                  (libcell)
A/                  (libpin)
B/                  (libpin)
Z/                  (libpin)

/libraries/cg/libcells/AO21A:
Total: 5 items
./                  (libcell)
A/                  (libpin)
B/                  (libpin)
C/                  (libpin)
Z/                  (libpin)
```

- The following command shows that the computed attribute `timing_case_computed_value` has been turned on:

```
legacy_genus:/>ls -computed /designs/violet/instances_comb/U1/pins_in/S
timing_case_computed_value = 1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following examples show the difference between the `ls -attribute` and `get_attribute` commands.

```
legacy_genus:/designs> ls -attribute
Total: 2 items
./
async_set_reset_flop_n/                 (design)
    Attributes:
      dft_mix_clock_edges_in_scan_chains = false
      wireload = /libraries/slow/wireload_models/sartre18_Conservative
```

```
legacy_genus:/designs> get_attribute wireload /designs/async_set_reset_flop_
n/
/libraries/slow/wireload_models/sartre18_Conservative
```

  The `ls -attribute` command lists all *user-modified* attributes and their values. The `get_attribute` command lists only the value of the specified attribute. The `get_attribute` command is especially useful in scripts where its returned values can be used as arguments to other commands.

- The following command lists all subdirectories in directory `sdp_groups`.

```
legacy_genus:/designs/test/sdp_groups> ls -R
./
ga/
ga/sdp_columns/
ga/sdp_datapaths/
ga/sdp_rows/
ga/sdp_rows/rb/
ga/sdp_rows/rb/sdp_columns/
ga/sdp_rows/rb/sdp_instances/
ga/sdp_rows/rb/sdp_instances/skip_instance_0
ga/sdp_rows/rb/sdp_instances/skip_instance_1
ga/sdp_rows/rb/sdp_instances/st_box1_g1
ga/sdp_rows/rb/sdp_instances/st_box2_g1
ga/sdp_rows/skip_row_0/
ga/sdp_rows/skip_row_0/sdp_columns/
ga/sdp_rows/skip_row_0/sdp_instances/
legacy_genus:/designs/test/sdp_groups>
```

Related Information

Related commands:

get_attribute on page 101

dirs on page 56

cd on page 53

popd on page 71

pushd on page 73

pwd on page 74

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

popd

`popd`

Removes the topmost element of the virtual directory (vdir) stack, revealing a new top element and changes the current vdir to the new top element. This command is similar to its UNIX counterpart.

Note: If `popd` is issued on a vdir stack that has only one element, an appropriate warning message is printed and the `popd` command exits without changing the vdir stack.

Examples

- In the following example, the vdir stack starts off as `/libraries /designs:` `/libraries` is the current vdir and the top element of the vdir stack, and `/designs` is next on the stack). When the `popd` command is issued, `/libraries` is popped off, and `/designs` becomes the top (and only element) of the stack and the current vdir.

```
legacy_genus:/libraries> dirs
/libraries /designs
```

```
legacy_genus:/libraries> popd
/designs
```

```
legacy_genus:/designs> dirs
/designs
```

```
legacy_genus:/designs> pwd
/designs
```

- In the following example, a `popd` command is issued on a vdir stack that has only one element.

```
legacy_genus:/> cd /designs
```

```
legacy_genus:/designs> dirs
/designs
```

```
legacy_genus:/designs> popd
vdir stack empty
/designs
```

This can happen when more `popd` commands are issued than `pushd` commands.

Related Information

Affects these commands:	dirs on page 56
	ls on page 67
	pushd on page 73

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

pwd on page 74

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

pushd

`pushd vdir`

Pushes the specified new target virtual directory (vdir) onto the vdir stack (as the topmost element) and changes the current vdir to that specified vdir. This command is similar to its UNIX counterpart.

Options and Arguments

vdir	Specifies the name of the vdir to be set as target vdir. You can use wildcards when they do not produce an ambiguous reference (more than one match).

Examples

- In the following example, `/libraries` is pushed onto the top of the `/designs` vdir stack and the current vdir is changed to it:

```
legacy_genus:/designs> pushd /libraries
/libraries /designs
legacy_genus:/libraries>
```

- In the following example, the push operation does not succeed because there is more than one vdir that starts with `ex`.

```
legacy_genus:/libraries> pushd ex*
Error   : A single object was expected, but multiple objects were found.[TUI-62]
        : The argument that found multiple objects was 'ex*'.
  pushd: push current dir onto stack and cd to new dir
Usage: pushd <object>
    <object>:
        new target vdir
Failed on pushd ex*
```

Related Information

Affects these commands:	dirs on page 56
	ls on page 67
	popd on page 71
	pwd on page 74

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

pwd

```
pwd
```

Displays the current position in the design hierarchy. This command is similar to its UNIX counterpart.

Examples

- The following example shows the current virtual directory (vdir) after changing the current vdir first:

```
legacy_genus:/> cd /designs
legacy_genus:/designs> pwd
/designs
```

Related Information

Related commands:

dirs on page 56
ls on page 67
popd on page 71
pushd on page 73

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_top_module

`set_top_module [design]`

Changes to the new top module of the specified design.

Options and Arguments

design	Specifies the name of the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

vdir_lsearch

`vdir_lsearch` *`list object`*

Performs an lsearch of a *vdir* type object in a list of objects. If a match is found, it returns the position of that object in the list. If no match is found, it returns `-1`.

While performing the search, the command performs a (fast) direct comparison of every vdir type object in the list with the pointer of the given object; whereas for every non-vdir type object in the list, it creates a string-name of the object and compares it with the string-name of the given object. This makes it faster than the regular lsearch command, which does string-name derivation and string comparison for every object in the list.

Note: Examples of commands that return vdir type objects are `find`, `edit_netlist` and `get_attribute`.

Options and Arguments

`list`	Specifies a list of objects. The objects can be regular objects or can be of type vdir.
`object`	Specifies the object for which you want to know the position in the list. The object must be of type `vdir`. **Note:** If the object is not of type `vdir`, a regular `lsearch` will be performed.

Example

In the following example, the first search, performed on a list of vdir type objects (result of `find` command), returns `-1`. To this list, two regular objects are added: `abc` and `/designs/test/instances_hier/mux_oo_5_10/pins_out/z`. The search on this mixed list of objects returns 4.

```
legacy_genus:/> set pin [get_attr driver /designs/test/nets/oo]
/designs/test/instances_hier/mux_oo_5_10/pins_out/z
legacy_genus:/> set list [find / -pin in*]
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_0
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_1
/designs/test/instances_comb/g4/pins_in/in_0
legacy_genus:/> vdir_lsearch $list $pin
-1
legacy_genus:/> lappend list "abc"
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_0
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
/designs/test/instances_comb/g4/pins_in/in_0 abc
legacy_genus:/> lappend list "/designs/test/instances_hier/mux_oo_5_10/pins_
out/z"
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_0
/designs/test/instances_hier/mux_oo_5_10/pins_in/in_1
/designs/test/instances_comb/g4/pins_in/in_0 abc
/designs/test/instances_hier/mux_oo_5_10/pins_out/z
legacy_genus:/> vdir_lsearch $list $pin
4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

vname

```
vname [instance | hnet | net | port_bus | pin_bus | pin | pgpin | port | subport
     | subdesign | design]...
```

Returns the Verilog name of the specified objects.

Note: You can specify a list of objects and the tool returns a list.

Example

The following command returns the Verilog name of instance `a_reg_3`.

```
vname [find / -inst a_reg_3]
DTMF_INST/DMA_INST/a_reg_3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

what_is

`what_is` *`object`*

Returns a string describing the type of object given as its argument. The command will work only if a single object is specified. A list of valid objects can be obtain by typing `find -help`.

This command is mostly used for writing Tcl scripts (rather than being an interactive command). It is useful for checking the types of arguments to the Tcl procedures.

Options and Arguments

`object`	Specifies the object for which you want to know the type.

Examples

- The following example returns pin:

```
what_is /designs/TOP/instances_hier/SUB/pins_in/A[0]
pin
```

- In the following example, the top-level design has two clocks `clock2` and `clock3`. The following command returns the type of `clock2`.

```
legacy_genus:/> what_is clock2
clock
```

- The following command fails because there is more than one object of the given name in the current tree structure.

```
legacy_genus:/designs/alu> what_is alu*
Error : A single object was expected, but multiple objects were found. [TUI-62]
      : The argument that found multiple objects was 'alu*'.
  what_is: return an object's type
Usage: what_is <object>
    <object>:
        drs object of interest
Failed on what_is alu*
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

what_is_list

```
what_is_list
     object...
```

Returns a list of strings describing the object types of the specified object(s).

Options and Arguments

object	Specifies the object whose types you want to know.

Examples

- The following command returns the object types for CLK1.

```
legacy_genus:/> what_is_list CLK1
exception cost_group
```

Using the `find` command returns the paths to these object types:

```
legacy_genus:/> find / * CLK1
/designs/top/timing/clock_domains/domain_1/CLK1
/designs/top/timing/exceptions/path_groups/CLK1 /designs/top/timing/cost_
groups/CLK1
```

Using the `what_is` command would fail because multiple object types are found for CLK1:

```
legacy_genus:/> what_is CLK1
Error   : A single object was expected, but multiple objects were found.
[TUI-62] [what_is]
       : The argument that found multiple objects was 'CLK1'.
 what_is: returns an object's type

Usage: what_is <object>

   <object>:
       object of interest
Failed on what_is CLK1
```

Related Information

Related command:	what_is on page 79

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

2

General

- ? on page 86
- alias on page 87
- apropos on page 88
- attribute_exists on page 90
- cdnshelp on page 91
- check_ccr on page 92
- clear on page 93
- clear_redline_terminal on page 94
- date on page 95
- enable_transparent_latches on page 96
- exec_embedded_script on page 98
- exit on page 100
- get_attribute on page 101
- get_liberty_attribute on page 103
- get_read_files on page 104
- help on page 105
- include on page 106
- is_attribute on page 107
- is_common_ui_mode on page 108
- lcd on page 109
- license on page 110

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- license checkin on page 111
- license checkout on page 112
- license feature on page 113
- license list on page 114
- license version on page 115
- lls on page 116
- lminus on page 117
- lpopd on page 118
- lpushd on page 119
- lpwd on page 120
- man on page 121
- more on page 123
- quit on page 125
- redirect on page 126
- reset_attribute on page 129
- resume on page 131
- search_doc on page 132
- set_attribute on page 133
- set_proc_verbose on page 135
- sh on page 136
- shell on page 137
- source on page 138
- stop_suspend on page 139
- string_representation on page 140
- suppress_messages on page 142
- suspend on page 144
- tcl_load on page 145

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- test_super_thread_servers on page 148
- unsuppress_messages on page 149

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

?

`? [command]...[> file]`

Provides help on the specified Genus commands.

Note: You can get help at any stage of the design.

Options and Arguments

command	Specifies the command for which you want help. If you do not specify a command name, you get a brief summary of all Genus commands.
file	Specifies the name of the file to which to write the help.

Examples

- The following example requests help for the `all_inputs` command:

```
legacy_genus:/> ? all_inputs
That command is:

    General
============================================================================
all_inputs  returns all the input ports.

Command details:

  all_inputs: returns all the input ports.

Usage: all_inputs [-design <design>]

    [-design <design>]:
        limits list of input ports to the specified top-level design
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

alias

`alias` *`alias_name command_name`*

Defines an alias for the specified name.

Options and Arguments

`alias_name`	Specifies the alias name.
`command_name`	Specifies the name of the command for which you want to create an alias. **Note:** Command name can also be specified with in " " or { }.

Example

The following command creates an alias ai for the `all_inputs` command.

```
legacy_genus:/> alias got get_object_type
got
legacy_genus:/> got -h
  get_object_type: returns an object’s type

Usage: get_object_type <object>+

    <object>+:
        object(s) of interest
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

apropos

```
apropos [-skip_help] [-skip_commands]
     [-skip_attributes] string
```

Performs a case insensitive wildcard search of commands (including their options) and attributes. The command will even encompass the help text of commands and attributes. The search results will be categorized into the following different sections:

- Commands that match search text
- Commands with help text that match search text
- Commands with options (both option name and option help text) that match search text
- Attributes that match search text
- Attributes with help text and attribute objects that matches search text

Options and Arguments

`-skip_help`	Do not include help text in the search results.
`-skip_commands`	Do not include commands in the search results.
`-skip_attributes`	Do not include attributes in the search results.
`string`	Specifies the search string

Examples

- The following example searches for the term `generic`:

```
legacy_genus:/> apropos generic
Commands matching search string:
  syn_generic

Commands with help text matching search string:
  syn_generic

Commands with option text matching search string:
  syn_generic
  write_hdl

Attributes matching search string:
  hdl_append_generic_ports (root)
  write_vlog_generic_gate_define (root)

Attributes with help text matching search string:
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
  allow_sharing_subdesign (subdesign)
  ...
```

- The following example searches for the term `generic` among commands only:

```
legacy_genus:/> apropos -skip_attributes generic
Commands matching search string:
  syn_generic

Commands with help text matching search string:
  syn_generic

Commands with option text matching search string:
  syn_generic
  write_hdl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

attribute_exists

```
attribute_exists attribute
     {-path object_paths | -type object_type}
```

Checks is the specified attribute exists.

A 1 is returned when the attribute exists, a 0 when the attribute does not exist.

Options and Arguments

attribute	Specifies the attribute name to be checked.
-path *object_path*	Specifies the path to the object for which the attribute must be checked.
-type *object_type*	Specifies the object type for which the attribute must be checked. **Note:** The command help will show a list of all valid object types.

Examples

- The following command checks whether `osc_source` exists for instances.

```
legacy_genus:/> attribute_exists osc_source -type instance
0
```

- The following command checks whether `count` exists for message TUI-20.

```
legacy_genus:/> attribute_exists count -path /messages/TUI/TUI-20
1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cdnshelp

```
cdnshelp
```

Launches Cadence Help viewer that displays product documentation organized in the order of Releases -> Products -> Manuals -> Topics.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_ccr

`check_ccr string`

Allows you to check if a fix for the specified CCR is *integrated* in the current or previous build.

- The command returns 1 if a fix for the specified CCR was integrated in this build or a previous build.
- The command returns 0 if a fix for the specified CCR was not integrated in this build or a previous build.

Notes

The command will only return 1 if the CCR was marked Checked_In or Validated at the time of the build.

You can only check for master CCRs. Duplicate CCRs are not included.

Options and Arguments

string	Specifies the CCR number to be checked.

Example

The following command indicates that the specified CCR was integrated in the build.

```
legacy_genus:/> check_ccr 1760464
1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clear

```
clear
```

Clears the terminal screen.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clear_redline_terminal

```
clear_redline_terminal
```

Clears the terminal screen.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

date

```
date
```

Returns the date and time. This command is equivalent to the UNIX `date` command.

Examples

```
legacy_genus:/> date
Thu Apr 23 01:28:55 PM PDT 2009
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

enable_transparent_latches

```
enable_transparent_latches
```

Enables transparent latches in the design by disabling the `EN` to `Q` arcs in the latch. This command must be used after `elaborate`. Transparent latches are latches with the enable signal held constant at the active state. Without enabling transparent latches, paths through them cannot be traced.

Related Information

Affects this command:	report disabled_transparent_latches on page 632

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

eval_common

`eval_common` *`script_name`*

Evaluates a Tcl script in the new Common UI mode.

This command is also aliased to `eval_common_ui`.

Options and Arguments

`script_name`	Specifies the name of the TCL script to evaluate.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

exec_embedded_script

```
exec_embedded_script
     [-design string] [-subdesign string]
```

Executes embedded scripts found in a specified design or subdesign. To execute the scripts on all top-designs and their subdesigns, use the `exec_embedded_script` command without any arguments. Use this command after the `elaborate` command.

The embedded script of a design or subdesign is stored in the `embedded_script` attribute of that design or subdesign.

Options and Arguments

`-design` *`string`*	Specifies the top level design for which the embedded script need to be executed. Using this option executes scripts for the specified design and every subdesign in it.
`-subdesign` *`string`*	Specifies the full path of the subdesign for which the embedded script needs to be executed. Using this option executes the script only for the specified subdesign.

Examples

- The following commands execute a SDC script, if any, embedded in the RTL description of the design.

  ```
  rc> exec_embedded_script
  rc> exec_embedded_script -design name of top design
  rc> exec_embedded_script -design full vdir path to top design
  rc> exec_embedded_script -subdesign full vdir path to subdesign
  ```

- If design or subdesign is not specified, then using this command executes the embedded SDC script of all designs and all subdesigns.
- If a design is specified, then this command executes the embedded SDC script in the RTL code of the given design and all its subdesigns.
- If a subdesign is specified, then this command executes the embedded SDC script in the RTL code of the specific subdesign only.

  Note: The impact of executing only this embedded script can still be hierarchical. For example, this can happen if an SDC command in the embedded script at this level of design hierarchy specifies a constraint for some instance down in the hierarchy.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- If both a design and a subdesign is specified, then the subdesign setting is ignored.

Related Information

Related command:	elaborate on page 505
Related attributes:	hdl_auto_exec_sdc_scripts

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

exit

`exit [code]`

Exits from the Genus shell without saving design data.

`Control-c` is a shortcut to the `exit` and `quit` commands.

Important

When exiting, no design data is saved, so it is important to save the design using the write_hdl and write_script commands.

Options and Arguments

code	Specifies the exit code. The following are the built-in exit codes: 0 – normal exit 1 – abnormal exit 246 – exit when no license server is available 245 – exit when no license feature is available 244 – exit when syntax error in script

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_attribute

```
get_attribute {attribute_name [object]
     | -h {attribute_name|*} {type|*} }
```

Retrieves the value of an attribute set by Genus or via the `set_attribute` command.

You can also use this command to list

- All attributes associated with a given object type
- All objects to which the specified attribute applies

This command is most commonly used within scripts to control the way a script operates based on the value of the attribute, or to retrieve basic information on the tool.

Options and Arguments

attribute_name	Specifies the name of an attribute whose value you want to retrieve.
object	Specifies the path to the object for which the attribute value should be retrieved. *Default*: current working directory
type	Specifies the object type for which you want the list of attribute names. Check the command help for a list of the valid object types. **Note:** Some of the object types currently do not have any attributes associated with them.

Examples

- The following example retrieves the current value of the `library` attribute on the root directory:

  ```
  legacy_genus:/> get_attribute library /
  tutorial.lbr
  ```

- The following example assumes you are already at the root of the design hierarchy, so the object specification is omitted:

  ```
  legacy_genus:/> get_attribute library
  tutorial.lbr
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example stores the attribute value in a variable:

```
legacy_genus:/> set old_library [get_attribute library /]
tutorial.lbr
```

- The following example returns the area of library cell:

```
legacy_genus:/> get_attribute area [find /lib* -libcell nor*]
1.5
```

- The following example lists all root-level attributes starting with `lp`:

```
legacy_genus:/> get_attribute lp* root -help
```

- The following example lists all attributes for all object types:

```
legacy_genus:/> get_attribute * * -help
```

- The following examples show the difference between the ls `-attribute` and `get_attribute` commands.

  - Using the `ls -attribute` command:

```
legacy_genus:/designs> ls -attribute
Total: 2 items
./
async_set_reset_flop_n/                 (design)
    Attributes:
      dft_mix_clock_edges_in_scan_chains = false
      wireload = /libraries/slow/wireload_models/sartre18_Conservative
```

  - Using the `get_attribute wireload` command:

```
legacy_genus:/designs> get_attribute wireload /designs/
async_set_reset_flop_n/
```

    returns the following wireload model:

```
/libraries/slow/wireload_models/sartre18_Conservative
```

    The `ls -attribute` command lists all user modified attributes and their values. The `get_attribute` command lists only the value of the specified attribute. The `get_attribute` command is especially useful in scripts where its returned values can be used as arguments to other commands.

Related Information

Affected by this command:	set_attribute on page 133
Related command:	ls on page 67

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_liberty_attribute

```
get_liberty_attribute attribute_name
     {libarc | libcell | libpin | pg_lib_pin | library | operating_condition
     | wireload}
```

Options and Arguments

attribute_name
: Specifies the name of the Liberty attribute whose value you want to retrieve.

libarc | *libcell* | *libpin* | *pg_lib_pin* | *library* | *operating_condition* | *wireload*
: Specifies the name of the object for which you want to retrieve a Liberty attribute value.

Example

The following command retrieves the `default_operating_conditions` attribute of the `tutorial` library.

```
legacy_genus:/> get_liberty_attribute default_operating_conditions tutorial
typical_case
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_read_files

```
get_read_files [-quiet]
     [-command command | -clear command]
```

Returns information on the files that have been read in.

Options and Arguments

`-clear command`	Removes the entries stored for the specified command.
`-command`	Specifies the command for which you want the information. By default, the command returns the information for all commands that read in files
`-quiet`	Suppresses the status message.

Example

```
legacy_genus:/> get_read_files
{source ./simple_ple.g} {read_netlist dtmf_chip.v} {read_sdc dtmf_tight.sdc}
{read_def fplan_mp.def}
legacy_genus:/> get_read_files -command read_def
fplan_mp.def
legacy_genus:/> get_read_files -clear read_sdc
Clearing recorded files for command 'read_sdc'.
legacy_genus:/> get_read_files
{source ./simple_ple.g} {read_netlist dtmf_chip.v} {read_def fplan_mp.def}
legacy_genus:/> get_read_files -clear source -quiet
legacy_genus:/>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

help

`help [command]...[> file]`

Provides help on the specified Genus commands.

Note: You can get help at any stage of the design.

Options and Arguments

command	Specifies the command for which you want help. If you do not specify a command name, you get a brief summary of all Genus commands.
file	Specifies the name of the file to which to write the help.

Example

- The following example requests help for the `include` command:

```
legacy_genus:/> help include
That command is:

  General
============================================================================
include reads in a command file

Command details:

  include: reads in a command file

Usage: include <string>

    <string>:
        the name of the file
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

include

`include` *`file`*

Executes scripts containing Genus or Tcl commands in the order they are listed in the `include` command.

When Genus begins executing, a number of scripts are automatically included.The use of script files allows automation and modularization of the design flow by placing commonly used commands and sequences of commands into their own scripts. For externally defined components like memories, the vendor can supply a script to create and set up the memory.

This command is identical to the `source` command.

Note: If an error occurs during execution of one of the scripts, execution of that script (and all scripts following it) is stopped.

Options and Arguments

`file`	Specifies the name of the script file to include.

Examples

- The following example includes one script file:

```
legacy_genus:/> include constraint.g
```

- The following example includes multiple scripts at once:

```
legacy_genus:/> include constraint.g; include synth.g
```

Related Information

Affected by these attributes:	init_hdl_search_path
	init_lib_search_path
	script_search_path

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

is_attribute

`is_attribute` *`attribute`* `-obj_type` *`object_type`*

Checks whether the specified attribute exists or not.

A `1` is returned when the attribute exists, a `0` when the attribute does not exist.

Options and Arguments

`attribute` Specifies the attribute name to be checked.

`-obj_type` *`object_type`*

Specifies the object type for which the attribute must be checked. All valid object types can be found by typing `get_db object_types .name` at the prompt.

Note: The command help will show a list of all valid object types.

Related Information

Related command: get_attribute on page 101

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

is_common_ui_mode

```
is_common_ui_mode
```

Checks if the common-UI mode is enabled or not.

Checks if the Stylus Common UI mode is enabled or not. This command returns "1" or "0" depending on whether it is executed in Stylus Common UI mode or not (respectively).

Examples

- The following example shows execution of `is_common_ui_mode` command on the Genus prompt:

  ```
  genus@root: > is_common_ui_mode
  1
  ```

  Return value 1 indicates that the common-UI mode is enabled.

  ```
  legacy_genus:/> is_common_ui_mode
  0
  ```

  Return value 0 indicates that the Stylus Common UI mode is not enabled. Genus prompt is also in legacy UI mode.

- The following example checks UI mode of current Genus prompt and executes required code in respective UI:

  ```
  if {[is_common_ui_mode]} {
      set_db syn_generic_effort medium
  } else {
      set_attr syn_generic_effort medium
  }
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lcd

`lcd` *`directory`*

Changes the UNIX working directory to the specified directory.

Options and Arguments

`directory`	Specifies the UNIX directory to which to change the current directory.

Examples

- The following example changes the working directory to the `rtl_lab01` directory in the current UNIX directory.

```
legacy_genus:/> lcd rtl_lab01
```

Related Information

Affects these commands:

lls on page 116
lpwd on page 120
shell on page 137

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license

```
license {checkin | checkout | feature | list | version}
```

Manages the checking in and checking out of licenses.

Options and Arguments

`checkin`	Checks in a product license that was previously checked out.
`checkout`	Checks-out additional licenses.
`feature`	Checks if the license is available.
`list`	Returns a list of licenses currently checked out.
`version`	Returns the checked out license version.

Related Information

Related commands:	license checkin on page 111
	license checkout on page 112
	license feature on page 113
	license list on page 114
	license version on page 115

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license checkin

`license checkin` *`license`*

Checks in a product license that was previously checked out.

Options and Arguments

`license`	Specifies the license to be checked in. The licenses that can be checked in are: ■ `Genus_CPU_Opt` ■ `Genus_Synthesis`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license checkout

`license checkout` *`license`* `[-version` *`float`*`] [-wait]`

Checks-out an additional product license. Licenses can only be checked-out one at a time.

If the license is available, it is checked out and `1` is returned. If the license is not available, a warning message is issued and `0` is returned. To check-in a license, use the `license checkin` command or quit Genus.

Options and Arguments

`license`	Specifies the license to be checked out. The licenses that can be checked out are: ■ `GEN_ENG100` ■ `Genus_Physical_Opt` ■ `Genus_CPU_Opt` ■ `Genus_Low_Power_Opt` ■ `Genus_Synthesis`
`-version` *`float`*	Allows you to specify a specific version of the license. By default, the command uses the hardcoded primary or alternate version of the license that you want to check out.
`-wait`	Waits until a license becomes available. The wait timeout is controlled by the value specified for the `-wait` option when you started Genus. The default wait timeout is 600s. **Note:** This option does not require the `-queue` option to be specified when starting Genus.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license feature

`license feature` *`license`* `[-version` *`float`*`]`

Checks if the specified license feature product license feature exists. The command returns “1” if the specified license feature is available, and “0” if it is not available.

Use this command in scripts to check a license before checking it out.

Options and Arguments

`license`	Specifies the license to be checked. The licenses that can be checked in are: ■ `GEN_ENG100` ■ `Genus_Physical_Opt` ■ `Genus_CPU_Opt` ■ `Genus_Low_Power_Opt` ■ `Genus_Synthesis`
`-version` *`float`*	Specifies the license version.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license list

```
license list
```

Returns a Tcl list of additional licenses that are currently checked-out. The license used to launch Genus is not included in this list.

Example

- The following example shows that one additional license is currently checked-out.

```
legacy_genus:/> license list
Genus_Low_Power_Opt
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

license version

`license version` *`license`*

Returns the actual version of the license checked out.

Options and Arguments

`license`	Specifies the license for which to check the version. You can specify any of the following: ■ `Genus_Physical_Opt` ■ `Genus_CPU_Opt` ■ `Genus_Low_Power_Opt` ■ `Genus_Synthesis`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lls

`lls` *`directory`*

Lists the contents of the specified UNIX directory.

Options and Arguments

`directory`	Specifies the UNIX directory whose contents to list.

Example

- The following example lists the contents of the `rtl_lab01` directory in the current UNIX directory.

```
legacy_genus:/> lls rtl_lab01
alu.v
lab01_gates.v
rc.cmd
rc.log
script.g
tutorial.lbr
legacy_genus:/>
```

Related Information

Affected by this command:	lcd on page 109
Related commands:	lpwd on page 120
	shell on page 137

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lminus

`lminus [-exact] list1 list2`

Removes one or more elements specified in the second list (list2) from the first specified list (list1). If none of the elements are found, genus will return original (list1).

Options and Arguments

`-exact`	Searches for an exact pattern match.
	By default, searches to expand wildcards.
`list1`	Specifies the list from which elements must be removed.
`list2`	Specifies the list of elements to remove from original list (list1).

Examples

- The following command subtracts the elements from list `l2` from list `l1`.

```
legacy_genus:/> set l1 {a  a* ab2  b1 b2 c}
legacy_genus:/> set l2 {a c}
legacy_genus:/> lminus $l1 $l2
a* ab2 b1 b2
```

- The following command subtracts a list consisting of `a*` and `c` from the list of elements of `l1`.

```
legacy_genus:/> lminus $l1 {a* c}
b1 b2
```

- The following command uses an exact search.

```
legacy_genus:/> lminus -exact  $l1 {a* c}
a ab2 b1 b2
```

- The following command return same list in case no element found:

```
legacy_genus:/> lminus $l1 {f}
a a* ab2 b1 b2 c
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lpopd

`lpopd`

Removes the topmost element of the UNIX directory stack, revealing a new top element and changes the current directory to the new top element. This command is equivalent to the UNIX `popd` command.

Note: If `lpopd` is issued on a directory stack that has only one element, an appropriate warning message is printed and the `lpopd` command exits without changing the directory stack.

Example

- In the following example, the synthesis directory is removed from the UNIX directory stack.

```
legacy_genus:/> lpwd
/home/ria/rc_ex/synthesis
legacy_genus:/> lpopd
/home/ria/rc_ex
```

Related Information

Affects these commands:

lls on page 116
lpushd on page 119
lpwd on page 120

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lpushd

`lpushd` *`directory`*

Pushes the specified new target directory onto the UNIX directory stack (as the topmost element) and changes the current directory to that specified directory. This command is equivalent to the UNIX `pushd` command.

Options and Arguments

`directory`	Specifies the name of the UNIX directory to be set as target directory. You can use wildcards when they do not produce an ambiguous reference (more than one match).

Example

- In the following example, the synthesis directory is pushed on top of the current UNIX directory.

```
legacy_genus:/> lpwd
/home/ria/rc_ex
legacy_genus:/> lpushd synthesis
synthesis
legacy_genus:/> lpwd
/home/ria/rc_ex/synthesis
```

Related Information

Affects these commands:	lls on page 116
	lpopd on page 118
	lpwd on page 120

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lpwd

```
lpwd
```

Returns the UNIX working directory.

Example

- The following example shows the current UNIX directory.

```
legacy_genus:/> lpwd
/usr3/verilog_labs
```

Related Information

Affected by this command:	lcd on page 109
Related commands:	lls on page 116
	shell on page 137

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

man

`man [section] { command_name | attribute_name | message_ID}...`

This command formats and displays the on-line manual pages for a command, attribute, or message ID. In order to access on-line manual pages outside Genus, you must first set your `MANPATH` environment variable to the following path:

```
$CDN_SYNTH_ROOT/share/synth/man_legacy
```

After setting the `MANPATH` variable, you can obtain manpages for commands, attributes and messages, from within Genus or within the Unix shell.

To access the on-line manual page of a subcommand, use the following specification:

`man command_name-sub_command`

For example:

```
man edit_netlist-ungroup
```

Options and Arguments

`section`

If you specify a section number, man only looks in that section of the on-line manual.

Commands are in section 1.
Attributes are in section 2.
Messages are in section 3.

In the absence of section number, man will display the first matching manual page found looking at section 1, 2, 3 in that order.

`{command_name | attribute_name | message_ID}`

Name(s) of the command, attribute, or message ID.

Examples

- The following examples display syn_generic command on-line manual pages:

```
man syn_generic
man 1 syn_generic
```

- The following examples display on-line manual pages for message TIM-101:

```
man TIM-101
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
man 3 TIM-101
```

- The following examples display command on-line manual pages first for retime command and then for retime attribute:

```
man 1 retime
man 2 retime
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

more

`more` *`command`*

Displays the output of the specified command one screen at a time.This command is similar to the UNIX more command.

If the output is several screens, the percentage of characters displayed so far is also shown at the bottom of the screen.

To scroll down the display one line at a time, press *Return.*

To display the next screen, press the SPACE bar.

To stop the display, press *q.*

Options and Arguments

`command`	Specifies the command whose output you want to display with the `more` command.

Examples

The following command displays one screen for the output of the `write_scandef` command.

```
legacy_genus:/> more write_scandef
```

```
VERSION 5.4 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN ria_test ;

SCANCHAINS 6 ;
  - AutoChain_1_seg1_test_clk1_falling
#     + PARTITION p_test_clk1_falling
#       MAXBITS 5
    + START PIN DFT_sdi_1
    + FLOATING
       out1_reg[4] ( IN SI ) ( OUT QN )
       out1_reg[5] ( IN SI ) ( OUT QN )
       out1_reg[6] ( IN SI ) ( OUT QN )
       out1_reg[7] ( IN SI ) ( OUT QN )
       out1_reg[8] ( IN SI ) ( OUT QN )
    + STOP DFT_lockup_g1 D
;
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
  - AutoChain_1_seg2_test_clk2_falling
#     + PARTITION p_test_clk2_falling
#       MAXBITS 5
    + START DFT_lockup_g1 Q
    + FLOATING
       out2_reg[4] ( IN SI ) ( OUT QN )
       out2_reg[5] ( IN SI ) ( OUT QN )
       out2_reg[6] ( IN SI ) ( OUT QN )
       out2_reg[7] ( IN SI ) ( OUT QN )
       out2_reg[8] ( IN SI ) ( OUT QN )
    + STOP DFT_lockup_g259 D
;

  - AutoChain_1_seg3_test_clk3_falling
#     + PARTITION p_test_clk3_falling
#       MAXBITS 5
    + START DFT_lockup_g259 Q
    + FLOATING
       out3_reg[4] ( IN SI ) ( OUT QN )
       out3_reg[5] ( IN SI ) ( OUT QN )
       out3_reg[6] ( IN SI ) ( OUT QN )
       out3_reg[7] ( IN SI ) ( OUT QN )
       out3_reg[8] ( IN SI ) ( OUT QN )
--More--(51%)
```

- The following command outputs the library check report results:

  ```
  more report checks -library
  ```

- The following command outputs the HDL lint check report results, displaying every message for every message ID, one message per line:

  ```
  more report checks -hdl_lint -detail
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

quit

`quit`

Exits from the Genus shell without saving design data.
Control-C is a shortcut to the `exit` and `quit` commands.

Important

When exiting, no design data is saved, so it is important to save the design using the write_hdl and write_script commands.

Options and Arguments

`code`	Specifies the exit code. The following are the built-in exit codes: 0 – normal exit 1 – abnormal exit 246 – exit when no license server is available 245 – exit when no license feature is available 244 – exit when syntax error in script

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

redirect

```
redirect [-append] [-tee] [-stderr]
     [-variable] [-mesg] target command
```

Redirect command performs the same function as traditional unix redirection operators (> & >>) and redirects the output to a file (by default) or variable by using -`variable` option. You can also directly use redirect characters/operators without explicit use of the redirect command. Any Tcl return value from command is returned by redirect. If redirect is used at the prompt, any echoing of the Tcl return at the prompt is not redirected. You can also use the redirect command to write the contents directly to a gzip compressed file by ending the file name with .gz.

Note: Messages generated by the command whose output you are redirecting, can be mixed with the command output. To limit the number of messages included, you can change the value of the `information level` root attribute.

It is also possible to use redirect characters similar to Unix usage, for Genus commands, but not for built-in Tcl commands. This usage works for any Innovus command that does not explicitly register > or >> as options (if > or >> are visible in the man page or -help output for the command, it is explicitly registered).

Usage of redirect characters and how to use & to indicate a Tcl var name are shown below:

> *name* - redirect stdout to the file *name*

>> *name* - redirect stdout and append to the file *name*

> &*name* - redirect stdout to the Tcl var *name*

>> &*name* - redirect stdout and append to the Tcl var *name*

Options and Arguments

`-append`	Appends stdout/stderr output to the specified file or Tcl variable.s
command	Specifies the command or Tcl code to execute as a quoted string or curly braces.
`-mesg`	Prevents messages from being suppressed.
`-stderr`	Redirects stderr also to the specified file or Tcl variable.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

target	Specifies the name of the file or variable to which to write the output.
-tee	Redirects output to both, the file or variable and the standard output (stdout).
-variable	Redirects the output to a Tcl variable.

Examples

- The following example sends the output generated by the `report gates` command to a file called `gates.rep`:

```
redirect gates.rep "report gates"
```

- The following example appends information to the existing `gates.rep` file:

```
redirect -append gates.rep "report gates"
```

- The following example sends the report to `stdout` and to a file on the disk:

```
redirect -tee gates.rep "report gates"
```

- The following example prevents output generated during reading of the script from being sent to the screen by sending it to `/dev/null`.

```
redirect /dev/null "include script.g"
```

- The following example stores the timing report in a variable $`rep_var`. You can use Tcl commands to manipulate or parse the variable.

```
redirect -variable rep_var "report timing"
```

- The following examples output a timing and a `scan_power` gzip compressed report file:

```
redirect file.gz "report timing"
redirect file.gz "report scan_power"
```

- The following command redirects the output to a new file timing.txt

```
redirect timing.txt "time_design -post_route"
```

- The following command appends the output to the file test.out.gz in compressed gzip format

```
redirect test.out.gz "source test.tcl" -append
```

- The following command redirects and appends to the file a.log

```
report_area >> a.log
```

- The following command redirects to the Tcl var a

```
report_area > &a
```

- The following command redirects and appends to the Tcl var a

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
report_area >> &a
```

- The following command allows redirecting output of multiple commands with single redirect command to a file or variable:

```
redirect out.txt {
       ==> puts "Hello"
       ==> echo "World" }

exec cat out.txt
       Hello
       World
```

- Redirect command allow to tee output to previous output target:

```
redirect -tee out.txt {
      ==> echo "Hello"
      ==> redirect out.txt -append {
      ==> echo "World"
      } }

exec cat out.txt
       Hello
       World
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_attribute

```
reset_attribute
     { attribute_name [object...] [-quiet]
     | -h  {attribute_name|*] {type|*}}
```

Resets attribute value to a pre-defined default value. It returns a single value when resetting a single attribute, a list of values when resetting same attribute on different object instances, or a human readable table for multiple attributes in the following format:

```
Attribute_name value
```

For example:

```
place_xxx true
add_filler true
```

Options and Arguments

attribute_name	Specifies the attribute that should be returned to its default value. The "*" wildcard character is supported.
object	Specifies the object for which the attribute values should be returned to the default values.
-quiet	Suppresses those messages that indicate which attributes and objects are being affected.
type	Specifies the object type for which you want the list of attribute names that can be reset. The "*" wildcard character is supported. Check the command help for a list of the valid object types.

Examples

- The following example specifies that all retiming attributes should be returned to their default values:

  ```
  legacy_genus:/> reset_attribute retime* *
  ```

- The following example specifies that all attributes on all sequential instances be returned to their default values:

  ```
  legacy_genus:/> reset_attribute * [find / -instance *seq/*]
  ```

- The following command lists all valid attributes that you can reset:

  ```
  legacy_genus:/> reset_attribute -h * *
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Both `type` and `attribute_name` accept wildcard strings.

Related Information

Related commands: get_attribute on page 101
set_attribute on page 133

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

resume

```
resume
```

Restarts the current Tcl process and brings back the regular `legacy_genus:/>` prompt. This command only works in conjunction with the `suspend` command. See the `suspend` command to see how these commands work together to stop and restart a Tcl process.

Related Information

Related commands:	stop_suspend on page 139
	suspend on page 144

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

search_doc

`search_doc` *`string`*

Searches the specified string(s) in the installed documentation hierarchy and on Cadence Customer Website (COS), and displays the results in CadenceHelp browser, which is also launched through the cdnshelp command.

Options and Arguments

`string`	Specifies the string that needs to be searched in the installed Genus Documentation set.

Related Information

Related command:	cdnshelp on page 91

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_attribute

```
set_attribute
     { attribute_name attribute_value [objects]
       [-quiet] [-lock]
     | -h  {attribute_name | *] {type|*}}
```

Sets the value of a specified attribute.

You can also use this command to list

- All attributes associated with a given object type and whose values you can set
- All objects to which a given attribute applies and whose value you can set

Attributes are placed on directory objects to control the way they are processed by Genus. They can also be used to control the synthesis process, the style of the generated code, and numerous other things. A complete list of all attributes is contained in the *Genus Attribute Reference for Legacy UI*.

Note: Not all attributes can be set. Attempting to set a *read-only* attribute returns an error. The *Genus Attribute Reference for Legacy UI* indicates whether an attribute is read-write or read-only.

Options and Arguments

attribute_name	Specifies the name of the attribute whose value you want to set.
attribute_value	Specifies the new attribute value. The value can be either a Boolean, integer, or string. A compound string (containing spaces) should be represented as a list using double-quotes or braces.
-lock	Locks the specified attribute's value so that it cannot be changed. The attribute becomes read-only.
objects	Specifies the path(s) to the objects. *Default*: current directory
-quiet	Suppresses those messages that indicate which objects are being affected. Alternatively, when setting an attribute on an object, an information message will not be printed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

type	Specifies the object type for which you want the list of attribute names. Check the command help for a list of the valid object types.

Examples

- The following example lists all valid attributes that you can set:

  ```
  legacy_genus:/> set_attribute * * -help
  ```

  Both `type` and `attribute_name` accept wildcard strings.

- The following example lists all valid attribute names that contain the string `dont`:

  ```
  legacy_genus:/> set_attribute *dont* * -help
  ```

- The following example sets the `information_level` attribute, which controls the verbosity of the tools, to the value of `5` and assumes the current directory for the path:

  ```
  legacy_genus:/> set_attribute information_level 5
      Setting attribute of root /: 'information_level' = 5
  ```

- In the following example, the path needed to be specified because `information_level` is a root attribute and would not have been found in the current path:

  ```
  legacy_genus:/designs/alu> set_attribute information_level 5 /
      Setting attribute of root /: 'information_level' = 5
  ```

- The following locks the technology library search path to `/home/Test/foo` by locking the `init_lib_search_path` attribute. For the rest of the session, the `init_lib_search_path` attribute becomes read-only.:

  ```
  legacy_genus:/> set_attribute -lock init_lib_search_path /home/Test/foo
  ```

Related Information

Affects these commands:	ls on page 67
	get_attribute on page 101

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_proc_verbose

```
set_proc_verbose
     {procedure_name [-quiet] |[-report]}
```

Controls the verbosity of a procedure or reports all verbose procedures.

Options and Arguments

procedure_name	Specifies the name of the procedure to be made verbose.
-quiet	Changes the procedure back to not being verbose.
-report	Reports procedures that are verbose.

Examples

- The following commands sets `myprocedure` as verbose:

  ```
  set_proc_verbose myprocedure
  myprocedure
  ```

- The following command reports `myprocedure` as a verbose procedure

  ```
  set_proc_verbose -report
  myprocedure
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

sh

`sh` *`command_string`*

Executes a UNIX shell command from the Genus shell.

When executing shell commands, Genus uses `/bin/sh`.

Important

Attempting to change the working directory for the Genus shell using `shell cd ..` is not possible because each `shell` command is executed in its own shell, and that shell is killed once the command is complete.

Options and Arguments

`command_string`	Specifies the UNIX command to execute. You can specify any valid UNIX command. A sequence of commands must be specified in the same string.

Examples

- The following example uses the *sh* command to get the current date:

```
legacy_genus:/> sh date
…
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

shell

`shell` *`command_string`*

Executes a UNIX shell command from the Genus shell.

When executing shell commands, Genus uses `/bin/sh`.

Important

Attempting to change the working directory for the Genus shell using `shell cd ..` is not possible because each `shell` command is executed in its own shell, and that shell is killed once the command is complete.

Options and Arguments

`command_string`	Specifies the UNIX command to execute. You can specify any valid UNIX command. A sequence of commands must be specified in the same string.

Example

- The following example uses the *sh* command to get the current date:

```
legacy_genus:/> shell date
…
```

Related Information

Affected by this command:	lcd on page 109
Related commands:	lls on page 116
	lpwd on page 120

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

source

```
source [-quiet] [-verbose]
     [-rsrc] [-rsrcid] [-echo] file
```

Reads a file and executes the commands in the file (scripts).

Note: The default behavior is different between Stylus UI and Legacy UI because the default value of `source_verbose` attribute is true in Stylus UI, while it is false in Legacy UI.

Options and Arguments

`-echo`	Echoes each line in the script.
`file`	Specifies the file to be sourced.
`-quiet`	Prevents line-by line information printing.
`-rsrc`	Genus does *not* support this option.
`-rsrcid`	Genus does *not* support this option.
`-verbose`	Prints line-by line information.

Related Information

Related attributes:	source_verbose
	source_verbose_info
	source_verbose_proc

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

stop_suspend

```
stop_suspend
```

Terminates the script execution when you have stopped the current Tcl process with the `suspend` command (which brings up the `legacy_genus-suspend:/>` prompt) and returns from the suspend command loop (brings back the regular `legacy_genus:/>` prompt).

This allows you to have access to the GUI to analyze any issues reported so far.

Related Information

Related commands:	resume on page 131
	suspend on page 144

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

string_representation

`string_representation` *`string`*

Converts a Tcl object into a string.

Options and Arguments

`string`	Specifies the object to be converted.

Example

- The following example shows the effect of manipulations on a Tcl list (list of object pointers) versus a string.

  In the first part of the example, you create a variable `list` that contains the pointers to the instances in the design that have `sub` in their name. The tool finds two instances. Next you remove one of the instances and print the content of the variable `list` again. Because the variable contains a list of pointers, the pointer to the second object is now replaced by `object_deleted`. When you rename the first object in the list and then print the content of the variable `list` again, the tool returns the pointer to the renamed instance and a pointer indicating that one object was deleted.

```
legacy_genus:/> set list [find / -inst *sub*]
legacy_genus:/> /designs/top/U1/instances_hier/sub1
    /designs/top/U2/instances_hier/sub2
legacy_genus:/> rm [find U2 -instance sub2]
legacy_genus:/> puts $list
legacy_genus:/> /designs/top/U1/instances_hier/sub1 object_deleted
legacy_genus:/> mv /designs/top/U1/instances_hier/sub1 mySub1
legacy_genus:/> puts $list
legacy_genus:/> /designs/top/U1/instances_hier/mySub1 object_deleted
```

  If you convert the Tcl list into a string before you perform all these manipulations, the content of the variable `list` will not be affected by the manipulations.

```
legacy_genus:/> set list [string_representation [find / -inst *sub*]]
legacy_genus:/> { /designs/top/U1/instances_hier/sub1
    /designs/top/U2/instances_hier/sub2 }
legacy_genus:/> rm [find U2 -instance sub2]
legacy_genus:/> mv /designs/top/U1/instances_hier/sub1 mySub1
legacy_genus:/> puts $list
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> { /designs/top/U1/instances_hier/sub1
    /designs/top/U2/instances_hier/sub2 }
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

suppress_messages

`suppress_messages [-number integer] [-quiet] message_id...`

Suppresses printing of the specified message in the log file.

Options and Arguments

message_id	Specifies the message IDs to be suppressed. The list can be in the form of Tcl list or space separated collection of multiple message IDs. Wild cards are accepted
-number *integer*	Prints the specified message only *n* number of times in the current session. The default value is 0.
-quiet	Suppresses those messages that indicate which attributes are being affected for the messages that are being suppressed.

Examples

- The following example suppresses the VLOG-1 and VLOG-2 messages:

```
legacy_genus:/> suppress_messages { VLOG-1 VLOG-2 }
```

- The following example shows the effect of the -quiet option:

  with the -quiet option:

```
Warning : Potential variable name conflict. [TUI-666]
        : A variable that controls the tool's behavior was set. The
'lbr_use_test_cell_seq_mux_scan' variable has the same name as an
internal variable.
  Setting attribute of root '/': 'init_lib_search_path' =
/home/rcap/regs/rc/dft/SMG/native/analysis/test2/LIB
```

  without the -quiet option:

```
Warning : Potential variable name conflict. [TUI-666]
        : A variable that controls the tool's behavior was set. The
'lbr_use_test_cell_seq_mux_scan' variable has the same name as an
internal variable.
  Setting attribute of message 'LBR-41': 'max_print' = 0
  Setting attribute of message 'LBR-146': 'max_print' = 0
  Setting attribute of message 'LBR-23': 'max_print' = 0
  Setting attribute of root '/': 'init_lib_search_path' =
/home/rcap/regs/rc/dft/SMG/native/analysis/test2/LIB
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: unsuppress_messages on page 149

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

suspend

```
suspend
```

Stops the current Tcl process and brings up the `legacy_genus-suspend:/>` prompt. Any commands or attributes that are issued during this time will not have access to the temporary variables from the suspended Tcl process. However, global variables and Tcl processes can still be accessed. Type `resume` to restart the process from where it was stopped.

The `suspend` command gives you flexibility in managing and debugging your scripts. You can insert multiple `suspend` commands in a script. The `suspend` command can also be used in nested scripts. When the script reaches a point where you have inserted the `suspend` command, it stops and restores access to the Genus prompt. You can then type any command required for debugging.

For example,

```
read_hdl -f $RTL_FILE_LIST
suspend
elaborate $TOP_MODULE_NAME
```

Whenever you want to resume your script, just type `resume` at the Genus prompt. If you do not want to resume your script and wants to come out of `suspend` mode, just type `stop_suspend`.

Note: Any commands that you enter after issuing the `suspend` command but before the `resume` command will not be included in Tcl's command history. However, these commands will be included in Genus's command editor history.

Related Information

Related commands:	resume on page 131
	stop_suspend on page 139

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

tcl_load

`tcl_load` *`fileName`* [*`packageName`* [*`interp`*]]

This command loads binary code from a file into the application's address space and calls an initialization procedure in the package to incorporate it into an interpreter.

Note: The Tcl command `load` must be renamed to `tcl_load` to avoid conflict with the Genus compiler `load` which is aliased to `read_hdl`.

Options and Arguments

`filename`	Is the name of the file containing the code. Its exact form varies from system to system but on most systems it is a shared library, such as a .so file under Solaris.
`packageName`	Is the name of the package, and is used to compute the name of an initialization procedure.
`interp`	Is the path name of the interpreter into which to load the package (see the `interp` manual entry for details). If this argument is omitted, it defaults to the interpreter in which the `load` (`tcl_load`) command was invoked.

Description

Once the file has been loaded into the application's address space, one of two initialization procedures will be invoked in the new code. Typically the initialization procedure will add new commands to a Tcl interpreter. The name of the initialization procedure is determined by *`packageName`* and whether or not the target interpreter is a safe one. For normal interpreters the name of the initialization procedure will have the form `pkg_Init`, where *`pkg`* is the same as *`packageName`* except that the first letter is converted to upper case and all other letters are converted to lower case. For example, if *`packageName`* is `foo` or `FOo`, the initialization procedure's name will be `Foo_Init`.

If the target interpreter is a safe interpreter, then the name of the initialization procedure will be `pkg_SafeInit` instead of `pkg_Init`. The `pkg_SafeInit` function should be written carefully, so that it initializes the safe interpreter only with partial functionality provided by the package that is safe for use by untrusted code. For more information on Safe-Tcl, see the `safe` manual entry.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The initialization procedure must match the following prototype:

```
typedef int Tcl_PackageInitProc(Tcl_Interp *interp);
```

The `interp` argument identifies the interpreter in which the package is to be loaded. The initialization procedure must return `TCL_OK` or `TCL_ERROR` to indicate whether or not it completed successfully; in the event of an error it should set the interpreter's result to point to an error message. The result of the `load` command will be the result returned by the initialization procedure.

The actual loading of a file will only be done once for each *`fileName`* in an application. If a given *`fileName`* is loaded into multiple interpreters, then the first load will load the code and call the initialization procedure; subsequent loads will call the initialization procedure without loading the code again. It is not possible to unload or reload a package.

The `load` command also supports packages that are statically linked with the application, if those packages have been registered by calling the `Tcl_StaticPackage` procedure. If *`fileName`* is an empty string, then *`packageName`* must be specified.

If *`packageName`* is omitted or specified as an empty string, Tcl tries to guess the name of the package. This may be done differently on different platforms. The default guess, which is used on most UNIX platforms, is to take the last element of *`fileName`*, strip off the first three characters if they are lib, and use any following alphabetic and underline characters as the module name. For example, the command load libxyz4.2.so uses the module name xyz and the command load bin/last.so {} uses the module name last.

If *`fileName`* is an empty string, then *`packageName`* must be specified. The `load` command first searches for a statically loaded package (one that has been registered by calling the `Tcl_StaticPackage` procedure) by that name; if one is found, it is used. Otherwise, the `load` command searches for a dynamically loaded package by that name, and uses it if it is found. If several different files have been loaded with different versions of the package, Tcl picks the file that was loaded first.

Bugs

If the same file is loaded by different fileNames, it will be loaded into the process's address space multiple times. The behavior of this varies from system to system (some systems may detect the redundant loads, others may not).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following is a minimal extension:

```
#include <tcl.h>
#include <stdio.h>
static int fooCmd(ClientData clientData,
    Tcl_Interp *interp, int objc, char * CONST objv[]) {
    printf("called with %d arguments\n", objc);
    return TCL_OK;
    }
    int Foo_Init(Tcl_Interp *interp) {
        if (Tcl_InitStubs(interp, "8.1", 0) == NULL) {
    return TCL_ERROR;
    }
    printf("creating foo command");
    Tcl_CreateObjCommand(interp, "foo", fooCmd, NULL, NULL);
    return TCL_OK;
}
```

When built into a shared/dynamic library with a suitable name (e.g. libfoo.so on Solaris and Linux) it can then be loaded into Tcl with the following:

```
# Load the extension
switch $tcl_platform(platform) {
    unix {
        load ./libfoo[info sharedlibextension]
    }
}

# Now execute the command defined by the extension
200
```

See Also

info sharedlibextension, Tcl_StaticPackage(3), safe(n)

Keywords

binary code, loading, safe interpreter, shared library

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

test_super_thread_servers

```
test_super_thread_servers
```

Returns status information on the machines used for super-threading.

Example

```
legacy_genus:/> set_attr super_thread_servers {rcl117 rcl118} /
  Setting attribute of root '/': 'super_thread_servers' = rcl117 rcl118
legacy_genus:/> test_super_thread_servers
Info    : Attempting to launch a super-threading server. [ST-120]
        : Attempting to Launch server 1 of 2.
        : RC is entering super-threading mode and is launching a CPU server
process. This is enabled by the root attribute 'super_thread_servers' or
'auto_super_thread'.
            : The server is rsh process '23453' to host 'rcl117'.
Info    : Attempting to launch a super-threading server. [ST-120]
        : Attempting to Launch server 2 of 2.
            : The server is rsh process '23459' to host 'rcl118'.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

unsuppress_messages

`unsuppress_messages` *`message_id...`*

Allows a previously suppressed message to be printed. The list can be in the form of TCL list or space separated collection of multiple message IDs. Wild cards are accepted.

Options and Arguments

`message_id`	Specifies the message identification. The list can be a TCL list or space-separated collection of message IDs. Wilcards are accepted.

Example

- The following example suppresses the `VLOG-1` and `VLOG-2` messages and then allows them to be printed again:

```
legacy_genus:/> suppress_messages { VLOG-1 VLOG-2 }
legacy_genus:/> unsuppress_messages { VLOG-1 VLOG-2 }
```

Related Information

Related command: suppress_messages on page 142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

3

Flowkit

Tip

For usage information, refer to the “Flow” chapter in the *Innovus Common UI Migration Guide*.

- check_flow on page 153
- create_flow on page 155
- create_flow_step on page 156
- define_feature on page 158
- delete_flow on page 160
- delete_flow_step on page 161
- edit_flow on page 162
- flowkit_template on page 163
- get_feature on page 165
- get_flow_config on page 166
- is_flow on page 167
- read_flow on page 168
- report_flow on page 169
- run_flow on page 170
- schedule_flow on page 172
- set_feature on page 173
- set_flow_config on page 174
- set_flowkit_read_db_args on page 175

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- set_flowkit_write_db_args on page 176
- write_flow on page 177
- write_flow_template on page 178

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_flow

```
check_flow
     [-fail string] [-flow string] [-proc string]
     [-step string] [-from string] [-to string]
```

Creates a single flow_step object and fills-in the required, and some optional attributes. Further modifications of the step can be performed using the `set_db` command. If you try to define a step with the same name as an existing step, it results in error. To redefine a flow step, you must use the `set_db` command to modify the attributes, or delete and recreate it.

Note: Unlike Tcl procedures, flow steps do not take arguments and refer to attributes instead. The use of attributes requires that the data on which a step relies, is stored in the database and may be examined later.

Options and Arguments

`-fail` *`string`*	Reports a problem. This is not used for analysis. It can only be used from Tcl called as part of static checking. Calling from outside static checking results in error.
`-flow` *`string`*	Runs the specified flow (created with the `create_flow` command). The `-from` and `-to` options are based on the flow steps. *Default*: `flow_step_order`
`-from` *`string`*	Specifies the name of the step to start from. Specify a step in the flow. By default, the flow starts from the first step that has not been run, or the first step if all steps have been run previously.
`-proc` *`string`*	Checks the specified procedure instead of analyzing a flow step.
`-step` *`string`*	Checks just the specified step. This option takes priority over the `-from` and `-to` options.
`-to` *`string`*	Specifies the name of the last step to be executed. By default, runs to the end of the last step in the flow. Specify a step in the flow, later than the `-from` step (if any).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_flow

```
create_flow -name name
     [-tool tool] [-tool_options options]
     [-owner string] steps
```

Creates a new, named flow object. `run_flow`, then allows the execution of the named flow (in addition to its existing functionality).

Options and Arguments

`-name` *`name`*	Specifies the name of the flow object to create.
`-owner` *`owner`*	Specifies the owner of the flow.
`steps`	Specifies the Tcl flow steps.
`-tool` *`tool`*	Specifies the name of the tool with which to run this flow. This option allows you to create flows for other tools than the one currently running. By default, the flow will be run in the current tool.
`-tool_options` *`options`*	Specifies the tool options to use when launching this flow.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_flow_step

```
create_flow_step
     -name name  body [-copy step] [-owner string]
     [-begin beginTcl] [-check checkTcl] [-end endTcl]
     [-categories string] [-write_db]
     [-skip_metric] [-exclude_time_metric]
```

Creates a single `flow_step` object and fills in the required and optional attributes. Further modifications of the step can be performed using the `set_db` command. If you try to define a step with the same name as an existing step, it results in error. To redefine a flow step, you must use the `set_db` command to modify its attributes, or delete the step and recreate it.

Note: Unlike Tcl procedures, flow steps do not take arguments and refer to attributes instead. The use of attributes requires that the data on which a step relies, is stored in the database and may be examined later.

Options and Arguments

`-begin` *`beginTcl`*	Specifies a block of Tcl to run at the start of the step.
`body`	Specifies a block of Tcl that is run as the main action of the step.
`-categories` *`string`*	Defines the metric categories to calculate the flow step.
`-check` *`checTcl`*	Specifies a block of Tcl to run step-specific static checks.
`-copy` *`step`*	Copies the specified step to this step.
`-end` *`endTcl`*	Specifies a block of Tcl to run at the end of the step.
`-exclude_time_metric`	Prevents that CPU time and wall time for this step are included in the parent steps.
`-name` *`name`*	Specifies the name of the step. The name can consist of alphanumeric characters and underscore only, and may not start with a digit. The following names have special meanings and cannot be used for step names: ■ next

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	■ end
`-owner` *`owner`*	Specifies the owner of the flow step.
`-skip_metric`	Skips generating metrics when the step is complete. By default, every step saves the design and runs metrics when completed.
`-write_db`	Automatically saves a database when the flow step is complete.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_feature

```
define_feature
     feature
     [-obj string] [-group string]
     [-description string]
     [-type {boolean|string|enum}]
     [-possible_values string]
     [-default string] [-required]
```

Defines a flow feature.

Options and Arguments

`-default string` Specifies the default value of the feature.

`-description string`

Specifies the detailed description of the flow feature. If it matches multiple features, then these features are listed along with a short description for each.

If it matches exactly one feature, a full description of this feature is printed. Matching no features results in an error.

feature Specifies the name of the feature.

`-group string` Outputs the exclusive group that contains features.

`-obj string` Specifies the flow or flow_step for a flow-specific feature. The features can be limited to a flow/step, or be global and usable in all the flows or steps.

Both the enabled and disabled code are included, with conditionals based on a new user attribute.

`-possible_values string`

Specifies the list of values that are valid for the features.

`-required` Indicates that this feature is mandatory.

`-type {boolean|string|enum}`

Specifies the type of feature value. The type can be boolean, a string, or an enumerated type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

Related commands: get_feature on page 165
set_feature on page 173

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_flow

```
delete_flow
     -name name
```

Deletes the flow that is cached in the database.

Options and Arguments

`-name` *`name`*	Specifies the name of the flow to be deleted.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_flow_step

```
delete_flow_step
     -name name
```

Deletes an existing `flow_step` object. If the step is present in the flow order, it is removed and an information message is displayed.

Options and Arguments

`-name` *`name`*	Specifies the name of the step to be deleted.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_flow

```
edit_flow [-append string] [-prepend string]
     [-reset] [-after string] [-before string]
     [-owner string] [-unique]
```

Edits the flow or flow_step objects.

Options and Arguments

`-after` *`string`*	Specifies to add the flow or flow_step objects at the end of the given flow or flow_step objects.
`-append` *`string`*	Specifies to append the flow or flow_step objects at the start or end of the given flow or flow_step objects.
`-before` *`string`*	Specifies to add the flow or flow_step objects at the start of the given flow or flow_step objects.
`-owner` *`owner`*	Specifies who edited the flow or flow step.
`-prepend` *`string`*	Specifies to prepend the flow or flow_step objects at the start or end of the given flow or flow_step objects.
`-reset`	Specifies to reset the customization of flow or flow_step objects at the start or end of the given flow or flow_step objects.
`-unique`	Specifies not to add the given flow or flow_step objects if they are already present on the given flow or flow_step objects..

Related Information

“Flow” chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

flowkit_template

```
flowkit_template
     [-describe string]
     [-directory string]
     [-enabled string]
     [-list_features]
     [-optional string]
     [-verbosity string]
```

Creates a set of template TCL files to be used as the basis for a flow. This command is used at the beginning after which, you can manage the generated and modified files. When invoking the command, you will select the type of flow required (implying a subset of template scripts to include) and the arrangement of sections across generated files.

On completion, a message is given to indicate creation of the flow template in the specified directory.

Options and Arguments

`-describe` *`string`*	Specifies detailed description of the available flow features. If it matches multiple features, then these features are listed along with a short description for each. If it matches exactly one feature, a full description of this feature is printed. Matching no features results in an error.
`-directory` *`string`*	Specifies the name of the directory for saving the generated files. If no directory is specified, the command uses the name flow. If required, the tool appends "_" and an integer to the directory name to avoid overwriting an existing directory. For example, flow_1, flow_2 and so on).
`-enabled` *`string`*	Specifies a list of flow features that are unconditionally incorporated into the generated scripts.
`-list_features`	Returns a TCL list of available flow features instead of generating templates.
`-optional` *`string`*	Specifies a list of flow features that are conditionally incorporated into the generated scripts. Both the enabled and disabled code are included, with conditionals based on a new user attribute.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-verbosity` *`string`* Sets the verbosity level, which determines the quantity of comments included in the generated files. Permissible values are:

- low—only include minimal comments
- high—include all comments
- default—include the default level of comments, which are explanatory but not discursive

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_feature

```
get_feature
     -feature string
     [flow | flow_step]
```

Returns the value of the specified flow feature.

Options and Arguments

`-feature string`	Specifies to name of the feature.
`flow | flow_step`	Specifies to name of the flow or flow step to which the feature applies. The features can be limited to a flow/step, or be global and usable in all the flows/steps.

Related Information

“Flow” chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_flow_config

```
get_flow_config
     string... [-quiet]
```

Returns the value for a key. If you specify multiple strings, the command interprets it as a set of keys that describe how to navigate the hierarchy of a data structure.

Options and Arguments

`-quiet`	Suppresses error messages if the specified key does not exist.
`string`	Specifies to name of a key in the data structure.

Examples

- The following command retrieves the value of the design_process_node.

  ```
  get_flow_config design_process_node
  ```

- The following command retrieves the library files of a library in a library set.

  ```
  get_flow_config library_sets slow_0v990 library_files
  ```

  This corresponds to looking in a data structure similar to:

  ```
  library_sets
      slow_0v990
          library_files {fill.lib pllclk_slow.lib}
          socv_files ...
      fast_0v792
          library_files ...
          socv_files ....
  ```

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

Related command: set_flow_config on page 174

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

is_flow

```
is_flow [-after string] [-before string]
     [-inside string]
     [-hierarchical]
     [-after_history string] [-quiet]
```

Queries if the current execution matches the specified condition. This helps you add actions depending on the context.

Options and Arguments

`-after` *`string`*	Specifies the current execution is after the flow or flow_step objects.
`-before` *`string`*	Specifies the current execution is before the flow or flow_step objects.
`-hierarchical`	Allows a search to go beyond the current flow boundary. This is done by flattening the entire flow execution graph into a linear chain of events (including all flow customization or scheduling). This enables to form more complex conditional statements especially when considering flows which consist of multiple sub-flows. The `-hierarchical` option allows before/after checks to be done on this type of structure.
`-inside` *`string`*	Specifies the current execution is within the flow or flow_step objects.
`-quiet`	Ensures that errors are reported for missing flow or flow_step objects.

elated Information

“Flow” chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_flow

`read_flow` *`string`*

Loads the content for the current flow and debugs the flow in the tool. The command accepts single argument as the name of a Tcl or yaml file.

Options and Arguments

`string`	Specifies the flow script file to read.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_flow

Refer to report_flow in the Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

run_flow

```
run_flow
     [-directory string]
     [-yaml string] [-flow string]
     [-no_check] [-no_summary] [-reset]
     [-step string] [-from string] [-to string]
```

Runs one or more flow steps by default, in the order specified in a single flow.

If more than one flow is defined, the `run_flow` command must be run with the `-flow` option. However, you can use the `-step` option, without that step being part of a flow.

Options and Arguments

`-directory` *`string`*	Specifies the working directory in which to run the flow. If a directory is not specified, the current directory is used. **Note:** Any relative paths in attributes such as `flow_report_directory` are relative to this directory.
`-flow` *`string`*	Runs the specified flow (created with the `create_flow` command). The `-from` and `-to` options are based on the flow steps. *Default*: `flow_step_order`
`-from` *`string`*	Specifies the name of the step to start from accepting a full dotted canonical path to a flow step object. By default, the flow starts from the step immediately after the last successfully run step (determined by examining the value of the `flow_history` root attribute), or the first step if that attribute is not set.
`-no_check`	Skips the error checks prior to running. By default, the checks are run.
`-no_summary`	Skips printing the full summary. By default, the summary is printed and mailed if the `flow_mail_to` root attribute is set.
`-reset`	Before running the command, it clears the flow history before running. After this, the flow starts from the first step if no explicit `-from` step is specified.
`-step` *`string`*	Runs just the specified step.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	This option takes priority over the `-from` and `-to` options.
`-to` *`string`*	Specifies the name of the last step to run to (inclusive; in other words, the named step is executed) accepting a full dotted canonical path to a flow step object. By default, runs to the end of the last step in the flow. Specify a step in the flow, later than the `-from` step (if any).
`-yaml` *`string`*	Specifies the `yaml` file to source before running the flow.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

schedule_flow

```
schedule_flow -flow string
     [-branch string] [-db string] [-directory string]
     [-no_db] [-no_sync | synch] [-defer]
```

Schedules a flow to be run after the current flow. The details of scheduled flows are stored in the `flow_schedule` attribute.

Options and Arguments

`-branch` *`string`*	Specifies the branch for the new (hierarchical) flow.
`-db` *`string`*	Specifies the starting database of the flow.
	If not supplied, the final database of the preceding flow (the one currently running) is used.
`-defer`	Specifies not to run the scheduled flow until requested.
`-directory` *`string`*	Specifies the working directory to use for the flow.
	By default, the current directory is used.
`-flow` *`string`*	Specifies the flow to be scheduled.
`-no_db`	When specified, flowtool does not use the database saved at the end of the step as the starting database for this flow.
`-no_sync`	When specified, flowtool does not wait for this flow to complete before continuing.
`-sync`	When specified, flowtool waits for all flows in the current branch to complete before starting.

Related Information

“Flow” chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_feature

```
set_feature
     -feature string
     [flow | flow_step]
     -value string
```

Specifies the value of the specified flow feature.

Options and Arguments

`-feature string`	Specifies to name of the feature.
`flow | flow_step`	
	Specifies to name of the flow or flow step to which the feature applies. The features can be limited to a flow/step, or be global and usable in all the flows/steps.
`-value string`	Specifies the value of the feature.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

Related commands:	define_feature on page 158
	get_feature on page 165

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_flow_config

```
set_flow_config
     string string...
```

Specifies the value for a key. If you specify multiple strings, the command interprets it as a set of keys with a single value, where the keys describe how to navigate the hierarchy of the data structure.

You can use these commands to set up the data structure to configure a flow.

Options and Arguments

string	Specifies to name of a key in the data structure.
string	The last string specified specifies the value for a key.

Examples

- The following command modifies the design_process_node.

  ```
  set_flow_config design_process_node 7
  ```

- The following command operates on an individual library set.

  ```
  set_flow_config library_sets slow_0v990 library_files {fill.lib
  pllclk_slow.lib}
  ```

  This corresponds to operating on a data structure similar to:

  ```
  library_sets
      slow_0v990
          library_files {fill.lib pllclk_slow.lib}
          socv_files ...
      fast_0v792
          library_files ...
          socv_files ....
  ```

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

Related command: get_flow_config on page 166

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_flowkit_read_db_args

`set_flowkit_read_db_args`

Sets the options for the `read_db` command when running the flow.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_flowkit_write_db_args

`set_flowkit_write_db_args`

Allows you to save databases through flow-kit in the Open Access format and specify the library or cell or view information. The `set_flowkit_write_db_args` command can be used to extend the options used to save a design on flow-steps, which are marked with `skip_db false`. This command needs to be placed inside the flow-step `body_tcl` of interest since it only affects that flow-step and resets back to default state when that step completes. The format of this command is similar to what would be used for `write_design`:

```
create_flow_step -name save {set_flowkit_write_db_args -rc -lib2ldb}
```

This saves a database named save, which uses the `-rc` and `-lib2ldb` options for `write_db`. Previous and subsequent flow-steps use default `write_design` options unless overridden using the `set_flowkit_write_db_args` command.

Note: This command supports all the parameters supported by `write_db`.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

"Overview" chapter in the *Innovus Common UI Migration Guide*

Related command: write_db on page 364

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_flow

```
write_flow
     [-filename string]
     [-include_state] [-directory string]
     [-flow_tcl] [-flow_yaml] [-setup_yaml]
```

Generates a Tcl script that may be sourced to reproduce the current flow. The script includes commands to:

- Set the value of all top-level attributes with non-default values to their current values
- Recreate all flow step objects with their current attributes, aside from those relating to run history (specifically, "`run_count`" and "`status`").

Sourcing this script reproduces (but not runs) the current flow in a fresh session, and forms a user-readable and editable record of the flow (compared to the flow data saved in the DB, which should not be read or edited directly).

Options and Arguments

`-directory` *`string`*	Specifies the directory in which all files must be written.
`-filename` *`string`*	Specifies the name of the file to generate.
`-flow_tcl`	Writes only the `flow.tcl` file in the directory.
`-flow_yaml`	Writes only the `flow.yaml` file in the directory.
`-include_state`	Writes out the current flow state. This can be used to restore the flow.
`-setup_yaml`	Writes only the `setup.yaml` file in the directory.

Related Information

"Flow" chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_flow_template

```
write_flow_template [-enable_feature string]
     [-optional_feature string]
     [-tools string] [-directory string]
     [-list] [-describe string]
     [-type string] [-version string]
     [-template_file string] [-tech_steps string]
```

Creates a set of template TCL files to be used as the basis for a flow.

Options and Arguments

`-describe` *`string`*	Specifies a detailed description of the available flow features of a template.
`-directory` *`string`*	Specifies the directory for saving the resulting flow.
`-enabled` *`string`*	Specifies the list of features required in the flow.
`-list`	Specifies the list of the available flow templates.
`-optional` *`string`*	Specifies the list of features that are optional in the flow.
`-tech_steps` *`string`*	Specifies the base recipe to overlay on the flow.
`-template_file` *`string`*	Specifies the flow template file.
`-tools` *`string`*	Specifies the list of tools whose scripts should be included in the flow.
`-type` *`string`*	Specifies the type of the flow to use
`-version` *`string`*	Specifies the version of the flow to use.

Related Information

“Flow” chapter in the *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

4

Metrics

Tip

For usage information, refer to the “Unified Metrics” chapter in the *Innovus Common UI Migration Guide*.

- create_metric_page on page 181
- create_metric_table on page 182
- create_metric_table_heading on page 184
- create_snapshot on page 185
- define_metric on page 187
- delete_metric on page 189
- delete_metric_page on page 190
- delete_metric_run on page 191
- delete_metric_table on page 192
- delete_metric_table_heading on page 193
- get_metric on page 194
- get_metric_alias on page 196
- get_metric_config on page 197
- get_metric_definition on page 198
- pop_snapshot_stack on page 199
- push_snapshot_stack on page 200
- read_metric on page 201
- read_metric_config on page 202
- report_metric on page 203

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- reset_metric_config on page 204
- reset_metric_run_id on page 205
- run_metric_category on page 206
- set_metric on page 207
- set_metric_alias on page 208
- set_metric_header on page 210
- set_metric_header on page 210
- update_metric on page 211
- write_metric on page 213
- write_metric_config on page 215

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_metric_page

```
create_metric_page -name string
     [-before string] [-after string][-format string]
```

Creates a new metric page.

Metrics are saved as part of the database and can be accessed using `get_attr` command.

Options and Arguments

`-after` *`string`*	Specifies the name of the page after which the new page should be inserted
`-before` *`string`*	Specifies the name of the page before which the new page should be inserted
`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-name` *`string`*	Specifies the name for the page. The page name should be unique.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_metric_table

```
create_metric_table -heading string
     -type (horizontal|vertical|cpu|dict|time_statement|histogram|imageplot)
     [-id table_id] [-auto_hide]
     [-per_snapshot] [-show_label_every frequency]
     [-hide_footers] [-hide_graph_footers] [-format string]
     [-hierarchy_separator separator] [-key string] [-collapsible_key]
```

Creates a metric table beneath a heading. The metrics html output contains many different types of elements for showing data, the `create_metric_table` command is used for configuring what is shown in the html output.

Options and Arguments

`-auto_hide`	Specifies whether to auto-hide the table or not. If specified, the table auto-hides, otherwise not.
`-collapsible_key`	Specifies to include collapse key columns in the table with the option to expand them.
`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-heading` *`string`*	Specifies the heading of the table. It is appended to the specified ID of the table.
`-hide_footers`	Specifies to hide the min, average and max rows from the bottom of the table. Graph buttons are still shown
`-hide_graph_footers`	Specifies to hide the graph buttons from the bottom of the table.
`-hierarchy_separator` *`separator`*	Specifies to show the metric names with hierarchy using the provided separator.
`-id` *`table_id`*	Specifies the ID for the new table.
`-key` *`string`*	Specifies a key for any regexp.
`-per_snapshot`	Specifies to expand the table per snapshot (`true` | `false`).
`-show_label_every` *`frequency`*	Specifies the frequency of X-axis labels in the histogram.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-type (horizontal | vertical | cpu | dict | time_statement | histogram | imageplot}`

Specifies the type of component to display data in the output.

`horizontal | vertical` - These tables allow the axes (rows/columns) to be swapped to change the orientation of displayed data.

`cpu` - A donut chart of CPU and wall-time usage.

`dict` - A table to show TCL-dict metrics data, generally used for capturing [report_obj].

`time_statement` - A simple line of text that shows the cpu/wall time used in current snapshot.

`histogram` - A component that shows histogram data.

`imageplot` - A component for showing image data (typically floorplans); can also merge multiple images (for example, show DRCs over floorplan).

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands:

create_metric_table_heading on page 184

delete_metric_table on page 192

delete_metric_table_heading on page 193

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_metric_table_heading

```
create_metric_table_heading -page page_name
     -title string [-id string] [-format string]
     [-type {fixed | collapse}]
```

Creates a metric table heading on a page which is created by the `create_metric_table` command.

Options and Arguments

`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-id` *`string`*	Specifies the ID for the new heading.
`-page` *`page_name`*	Specifies the page name to add the heading to.
`-title` *`string`*	Specifies the title to be shown in the heading.
`-type {fixed | collapse}`	Specifies the type of heading element, fixed (standard heading in the flow of the metrics html output) or collapse (a button containing a table, which is hidden by default). **fixed** - A fixed heading is a heading in the flow of the metrics html output. **collapse** - A collapse heading is a button which contains a table, which is hidden by default. This is used for detail numbers in a table without showing all the detail initially.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands:

create_metric_table on page 182
delete_metric_table on page 192
delete_metric_table_heading on page 193

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_snapshot

```
create_snapshot {-name string | -label string}
     [-auto {default|min}] [-exclude_time_metric]
     [-categories {all|{check|design|flow|hold|power|route|setup}...]
     [pattern] [-push] [-pop]
```

Creates a snapshot of the metric values.

All commands that save metrics call `create_snapshot` to save the metrics in a consistent manner. When `create_snapshot` is called, it creates a named or labeled record for the metrics in the database. The label for super commands are the command names.

Metrics are saved as part of the database and can be accessed using `get_attr` command.

Metric values should be saved such that the unit is easily known or accessible.

Options and Arguments

`-auto {default|min}`

Specifies which set of auto metrics to run. You can choose between the default set or a minimum set.

`-categories {all|check|design|flow|hold|power|route|setup}`

Specifies the metrics categories to compute.

If you specify all, the command selects the categories based on the current state of the design.

You can also specify a blank-separated list of categories.

`-exclude_time_metric`

Prevents that CPU time and wall time for this step are included in the parent snapshot calculation.

`-label` *`string`* Specifies the snapshot label.

`-name` *`string`* Specifies the name of the snapshot.

You can specify either a label or a name for the snapshot.

`pattern` If specified, it is the pattern which should be compared with the metrics. Only the metrics matching the pattern are included in the snapshot.

`-pop` Specifies to pop the snapshot stack and create a snapshot.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-push`	Specifies to push the snapshot onto the snapshot stack.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands:

define_metric on page 187
pop_snapshot_stack on page 199
push_snapshot_stack on page 200
read_metric on page 201
run_metric_category on page 206
set_metric on page 207
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_metric

```
define_metric -name metric [-type string]
     [-tcl tcl_command_list]
     [-inheritance_tcl string] [-no_inheritance]
     [-inherit_no_children]
     [-description string]
     [-units string] [-precision num_dec_places]
     [-compare function] [-threshold units]
     [-warning string]
     [-warning string]
```

Defines a new metric to be calculated.

To create the new metrics, you need to provide the TCL code. It can be a set of commands or a single procedure that returns a value.

The metric is calculated and stored with the other metrics using the `create_snapshot` command. All `define_metric` commands are saved and stored in the design database.

Options and Arguments

`-compare` *`function`* Specifies the compare function.

`moreBetter`

`lessBetter`

`moreBetterIgnorePositive`

`-description` *`string`*

Specifies some description for the metrics to be defined.

`-inherit_no_children`

Specifies to run the inheritance Tcl even if there are no children values.

`-inheritance_tcl` *`string`*

Specifies the Tcl for combining children values, equivalent proc args is {children}.

`-name` *`metric`* Specifies the name of the user-defined metric.

`-no_inheritance` Disables inheritance of the metric.

`-precision` *`num_dec_places`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the number of decimal places to be used for precision.

`-tcl` *`tcl_command_list`*

Specifies the Tcl commands or procedure to execute to generate the metric.

It is of the format {*`value unit`*}

`-threshold` *`units`* Specifies the threshold for numeric metrics.

`-type` *`string`* Specifies the type of the metric being defined.

`-units` *`string`* Specifies the units for reporting the metrics.

`-warning` *`string`* Specifies the warning threshold or name of the metric containing the threshold value.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Affects this command:	report_metric on page 736
Related commands:	define_metric on page 187
	delete_metric on page 189
	delete_metric_run on page 191
	read_metric on page 201

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_metric

```
delete_metric
     -name metric
```

Removes a previously user-defined metrics from all labels.

Options and Arguments

`-name` *`metric`*	Specifies the name of the user-defined metric to be removed.

Example

The following example removes the user-defined metric `state`.

```
delete_metric -name state
```

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Affects this command:	report_metric on page 736
Related command:	define_metric on page 187
	define_metric on page 187
	delete_metric_run on page 191

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_metric_page

`delete_metric_page -name` *`page_name`* `[-format` *`string`*`]`

Deletes a metric page created by the `create_metric_page` command.

Options and Arguments

`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-name` *`page_name`*	Specifies the name of the page to be deleted.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command: create_metric_page on page 181

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_metric_run

```
delete_metric_run
     -name metric
```

Deletes a metric run.

Options and Arguments

`-name` *`metric`*	Specifies the name of the metric run to be deleted.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Affects this command:	report_metric on page 736
Related command:	define_metric on page 187
	define_metric on page 187
	delete_metric on page 189

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_metric_table

`delete_metric_table -id` *`table_id`* `[-format` *`string`*`]`

Deletes a metric table created by the `create_metric_table` command.

Options and Arguments

`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-id` *`table_id`*	Specifies the ID of the table to be deleted.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands:

create_metric_table on page 182
create_metric_table_heading on page 184
delete_metric_table_heading on page 193

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_metric_table_heading

`delete_metric_table_heading -id` *`heading_id`* `[-format` *`string`*`]`

Deletes a metric page based on the heading ID.

Options and Arguments

`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-id` *`heading_id`*	Specifies the ID of the heading to be deleted.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_metric

```
get_metric [-id string] [-names string]
     [patternlist] [-exclude_inherited]
     [-pending] [-raw]
```

Gets the current value for a metric. This command is used to query the value of a metric from the current specified list of metrics as well as from any previous state.

Options and Arguments

`-exclude_inherited`	Specifies not to return any inherited value.
`-id` *`string`*	Specifies the ID of the database to output. ID is given by the `read_metric` command. *Default*: current database.
`-names` *`string`*	Specifies a list of pattern that matches the snapshot names. Wildcards are supported in the pattern. If not specified, it indicates current snapshot.
`patternlist`	Specifies a list of pattern to match the metrics. Wildcards are supported. Use * to get all metrics.
`-pending`	Specifies to return the metrics from the pending snapshot. This snapshot is not yet created.
`-raw`	Specifies to return the metrics as raw values. These are not normalized to defined units.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

read_metric on page 201
set_metric on page 207
set_metric_header on page 210

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_metric_alias

`get_metric_alias`

Returns the list of a metric aliases created by the `set_metric_alias` command.

Related Information

"Unified Metrics" chapter in the *Innovus Common UI Migration Guide*

Related command: set_metric_alias on page 208

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_metric_config

```
get_metric_config [-type {user|default|rnd}]
```

Gets the current configuration of a metric as a Tcl list or a dict.

Options and Arguments

`-type {user|default|rnd}`
: Specifies the type of the metric configuration.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

read_metric on page 201
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_metric_definition

`get_metric_definition [-id string]-name metric_name`

Gets the metric definition for the specified metric.

Options and Arguments

`-id string`	Specifies the database ID to output. ID is given by the `read_metric` command.
`-name metric_name`	Specifies the name of the metric.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

read_metric on page 201
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

pop_snapshot_stack

```
pop_snapshot_stack
```

Pops the metrics saved since the last push.

The `pop_snapshot_stack` command works in conjunction with `push_snapshot_stack` to create metric hierarchies. Any metrics captured inside a push/pop must be saved via create_snapshot (implicit or explicit) or they will be lost.

Examples

- The following example shows how to use the `pop_snapshot_stack` command:

```
push_snapshot_stack
syn_opt
pop_snapshot_stack
create_snapshot -categories all -label syn_opt
report_metric -format html -file syn_opt.html
```

Related Information

"Unified Metrics" chapter in the *Innovus Common UI Migration Guide*

Related command: push_snapshot_stack on page 200
create_snapshot on page 185
report_metric on page 203

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

push_snapshot_stack

`push_snapshot_stack [-name string]`

Creates a hierarchy for metric capture. All metrics captured inside the hierarchy is saved at this level during snapshot creation (implicit or explicit). Snapshot hierarchies can be nested so any number of hierarchies are supported.

Note: Every `push_snapshot_stack` command should have a corresponding `pop_snapshot_stack` command.

Options and Arguments

`-name` *`string`*	Specifies the name of the snapshot.

Examples

- The following example shows how to use the `push_snapshot_stack` command:

```
push_snapshot_stack
syn_opt
pop_snapshot_stack
create_snapshot -categories all -label syn_opt
report_metric -format html -file syn_opt.html
```

Related Information

"Unified Metrics" chapter in the *Innovus Common UI Migration Guide*

Related command:

pop_snapshot_stack on page 199
create_snapshot on page 185
report_metric on page 203

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_metric

```
read_metric filename -id string
     [-previous string] [-merge]
```

Loads the metrics file for querying, reporting and comparing. You can load any number of these files but each file should have a unique ID.

Options and Arguments

filename	Specifies the name of the file to read.
-id *string*	Specifies the name of the snapshot to create or overwrite.
-merge	Specifies to merge the metrics into the named snapshot instead of replacing its content.
-previous *string*	Specifies the unique ID of previous snapshots for new pending snapshot.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

get_metric on page 194
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_metric_config

```
read_metric_config
     -format string string
```

Reads the metric configuration file.

Options and Arguments

`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`string`	Specifies the filename from which to read the metric configuration.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

get_metric on page 194
set_metric on page 207
set_metric_header on page 210
write_metric on page 213
write_metric_config on page 215

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_metric

Refer to report_metric in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_metric_config

```
reset_metric_config [-default]
```

Resets the metric page configuration back to default.

Options and Arguments

`-default`	Specifies to set the contents back to default.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

get_metric on page 194
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_metric_run_id

```
reset_metric_run_id
```

Clears the current metric run ID.

When you are issue the create_snapshot command and send metrics data to a server, the job has a run_id to identify the data. If you load a database from another user, or want to perform experimentation, the reset_metric_run_id command allows you to clear the run_id and any related state so that it is sent with a new run_id.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command: read_metric on page 201

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

run_metric_category

`run_metric_category [-category category]`

Calculates the metrics for the specified category.

The metric values are saved in the database when you issue a `create_snapshot` command.

Options and Arguments

`-category category`

Specifies the name of the category to run.

The category can be any of `{check|clock|design|flow|hold|power|route|setup}`

When no category is specified, the `design` category is used by default. This is in alignment with the `create_snapshot` command.
This command is not required to be run by the user.
This command is used by the `create_snapshot` command to compute the metrics for each category.
But, the difference is `run_metric_category` leaves the computed metrics as "pending", so they will get added to the next snapshot which allows them to be combined with other pending metrics in the same snapshot.

Note: The `hold`, `cts`, and `route` categories do not compute any metrics in Genus.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command: create_snapshot on page 185

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_metric

```
set_metric -name string
     -value string
```

Sets a value for the specified metric.

Options and Arguments

`-name` *`string`*	Specifies a name for the metric.
`-value` *`string`*	Specifies the metric value.

Examples

- The following example sets the value of the worst negative setup slack (all path groups) to 0.100ns:

  ```
  set_metric -name timing.setup.wns -value
  {0.100ns}
  ```

- The following example sets the value of the total area of the design to 100 um:

  ```
  set_metric -name design.area -value {100 um}
  ```

Related Information

"Unified Metrics" chapter in the *Innovus Common UI Migration Guide*

Related commands: define_metric on page 187
set_metric_header on page 210

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_metric_alias

```
set_metric_alias
     -from old_name -to new_name
     [-no_overwrite]
     [-order order_of_priority]
```

Renames a metric name, prefixed with the name of an analysis view, path group, and so on, to another name. For example, you can rename `path_group:reg2reg` to `reg2reg` using the `set_metric_alias` command.

Options and Arguments

`-from` *`old_name`*	Specifies the metric object name to be changed.
`-no_overwrite`	Prevents overwriting of an already set alias.
`-order` *`order`*	Specifies the order of priority of metric object in tables.
`-to` *`new_name`*	Specifies the alias name for the metric object.

Example

- The following command renames `path_group:reg2reg` to `reg2reg` with priority order 1.

  ```
  set_metric_alias -from path_group:reg2reg -to reg2reg -order 1
  ```

- The following command renames `path_group:reg2cgate` to `reg2cgate` with priority order 2.

  ```
  set_metric_alias -from path_group:reg2cgate -to reg2cgate -order 2
  ```

- The following command queries the set of aliases:

  ```
  get_metric_alias
  ```

  Output

  ```
  path_group:reg2reg {to reg2reg order 1} path_group:reg2cgate {to reg2cgate
  order 2}
  ```

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands: get_metric_alias on page 196

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_metric on page 207

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_metric_header

```
set_metric_header -name string
     -value string
```

Sets the metric header.

Options and Arguments

`-name` *`string`*	Specifies a name for the metric.
`-value` *`string`*	Specifies the metric value.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command: set_metric on page 207

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

update_metric

```
update_metric -table string [-metric string] [-header string]
     [-title string] [-group string]
     [-graph_type (treemap | line | none)] [-show_graph]
     [-link_file_metric metric_name] [-link_table table]
     [-renderer html_format] [-color_met_target] [-format string]
```

Updates a metric table by adding a metric or a heading.

Options and Arguments

`-color_met_target`	Shows color in the table when the target is met.
`-format` *`string`*	Specifies the configuration for the `report_metric` format. Options: `{html excel text}`
`-graph_type (treemap|line|none)`	Specifies the type of graph to be used in the table. *Default*: `line`
`-group` *`string`*	Specifies the title of the group to be used in the table.
`-header` *`string`*	Specifies the name of the header to add to the table.
`-link_file_metric` *`metric_name`*	Specifies the name of the metric containing the filename to open when the metric value is clicked.
`-link_table` *`table`*	Specifies the table to scroll to when the metric value is clicked.
`-metric` *`string`*	Specifies the name of the metric to add to the table.
`-renderer` *`html_format`*	Forces a specific HTML format renderer for the histogram metric.
`-show_graph`	Specifies whether to show this metric in a comparison graph.
`-table` *`string`*	Specifies the ID of the table to add the metric to.
`-title` *`string`*	Specifies the title to be used in the table.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related commands:

get_metric on page 194
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_metric

```
write_metric [-format string]
     [-id string] [-names string]
     [metric_names] [-file string]
     [-inline] [-merge]
```

Writes out the current metrics to the specified file or stdout.

The `write_metric` command is used by `write_db` to write the metrics in the json format in the database directory. Super commands use `write_metric` to write metrics into the log file (stdout). This can also be used to write metrics in various formats by the user.

Options and Arguments

`-file` *`string`*	Specifies the name of the file to write.
`-format` *`string`*	Specifies the format of the data for the return value. Supported formats: `json`, `xml`, `csv`, `tcl`. *Default*: `json`
`-id` *`string`*	Specifies to output the database ID.
`-inline`	Specifies to return the result to stdout instead of writing it to the file.
`-merge`	Specifies to merge the results to an existing file.
`metric_names`	Specifies a list of patterns to match the metrics. *Default* is all metrics (*).
`-names` *`string`*	Specifies a list of patterns to match snapshot names.

Examples

- The following code shows the usage of the `write_metric` command:

```
# ---------------------------------------------------------------------
# Generate a XML file for all timing, power, and flow metrics from
# from prects to postcts for a previous run (id v101)
# ---------------------------------------------------------------------
read_metric -id v101 run101/dbs/postcts.enc.dat/design.metrics
write_metric -id v101 -format xml
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command: read_metric on page 201
set_metric on page 207
set_metric_header on page 210

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_metric_config

```
write_metric_config
     -format string string
```

Writes out the metric configuration file.

Options and Arguments

`-format` *`string`*	Specifies the format of the data for the return value. Supported formats: `json`, `xml`, `csv`, `tcl`. *Default*: `json`
`string`	Specifies the filename to which to write the metric configuration.

Related Information

"Unified Metrics" chapter in *Innovus Common UI Migration Guide*

Related command:

get_metric on page 194
read_metric_config on page 202
set_metric on page 207
set_metric_header on page 210
write_metric on page 213

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

5

GUI Text

Important

Most of the GUI commands are available in both the new GUI and legacy GUI. A note was added for commands or options that only apply to the legacy GUI.

- General GUI Text Commands on page 218
- HDL Viewer GUI Text Commands on page 225
- Schematic Viewer GUI Text Commands on page 229
- Physical Viewer GUI Text Commands on page 236

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

General GUI Text Commands

- gui_balloon_info on page 219
- gui_hide on page 219
- gui_info on page 219
- gui_legend on page 220
- gui_raise on page 221
- gui_reset on page 222
- gui_resume on page 222
- gui_selection on page 222
- gui_show on page 223
- gui_status on page 223
- gui_suspend on page 224
- gui_update on page 224

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_balloon_info

`gui_balloon_info`

Retrieves the text from the last balloon info.

Note: This command is only available in the legacy GUI.

gui_hide

`gui_hide`

Hides the GUI. Type the `gui_show` command to re-display the GUI.

Related Information

Related commands:	gui_raise on page 221
	gui_show on page 223

gui_info

`gui_info` *`string`*

Adds the specified text string in the info section of the status bar. The text remains until it is overwritten.

This command is usually used to print persistent messages (for example, Design is mapped).

Note: This command is only available in the legacy GUI.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_legend

`gui_legend -title` *`titlename`* `[-cancel` *`string`*`]` *`string`*

Creates a GUI legend. This is useful if you need a simple legend to color code information.

Note: You can use this command with the Physical Viewer and Schematic Viewer.

Options and Arguments

`-cancel` *`string`*	Specifies a Tcl proc that can be run when the Close button is pressed.
`string`	Specifies the data set to display in the legend.
`-title` *`titlename`*	Specifies the legend dialog title.

Example

- The following commands show how to create a legend.

  The first set of commands create the data set for the legend:

  ```
  set entry1 [list red   "This is entry 1"]
  set entry2 [list green "This is entry 2"]
  set entry3 [list blue  "This is entry 3"]
  set data [list $entry1 $entry2 $entry3]
  ```

  The next command specifies the TCl procedure to run when the Close button is pressed.

  ```
  proc foo {} {
     puts "Legend cancel"
  }
  ```

  The last command specifies the data set and title to display in the legend, and the procedure to run when the Close button is pressed:

  ```
  gui_legend -title "Legend Example" -cancel foo $data
  ```

  The following figure shows the resulting GUI legend.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Figure 5-1 GUI Legend

gui_raise

```
gui_raise [-nosync]
```

Keeps the GUI window on top of all other windows.

It can be also used to reopen the GUI main window hidden by gui_hide command or by clicking "Minimize Window" icon at the top-right corner. The functionality is very similar to gui_show command. The minor difference is that the gui_raise command reopens the Schematic Main window (Design Browser : "Schematic View" -> "In Main" menu) as well as the GUI Main window.

Options and Arguments

`-nosync`	Does not synchronize the GUI when the GUI is raised. **Note:** This option is only available in the legacy GUI.

Related Information

Related commands:	gui_hide on page 219 gui_show on page 223

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_reset

```
gui_reset
```

Resets the busy indicator if it remains busy.

The busy indicator can remain red if the script that set the busy indicator is broken and therefore the busy indicator does net get reset or cleared.

gui_resume

```
gui_resume
```

Resumes automatic GUI updates.

Note: This command is only available in the legacy GUI.

Related Information

Related command: gui_suspend on page 224

gui_selection

```
gui_selection
```

Returns the selection list of the currently selected objects in the GUI. The list consists of the object path for instance, net, pin, and port objects. The list can contain multiple objects.

Note: This command is only available in the legacy GUI.

Example

- The following example returns a combinational instance:

  ```
  legacy_genus:/> gui_selection
  /designs/mul_clk/instances_comb/g14
  ```

  If you enable the toolbar by un-checking *Hide Toolbar* under the Preferences menu, the complete path for the last selected object will be displayed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_show

`gui_show [-nosync]`

Displays the GUI main window if `gui_hide` command was used to hide it or for the first time display of GUI when the tool is launched.

If you run the `gui_show` command when the Genus main window is already displayed, the tool retains the current zoom level and displays the main window without refreshing it.

You can run the gui_hide command to hide the Genus main window.

The functionality of this command is similar to that of gui_raise command, difference being:

1. `gui_show` displays/reopens only the GUI main window and not the schematic main window.
2. It does not place the main window on top of other open windows.

Options and Arguments

`-nosync`	Does not synchronize the GUI when the GUI is raised. **Note:** This option is only available in the legacy GUI.

Related Information

Related commands:	gui_hide on page 219 gui_raise on page 221

gui_status

`gui_status` *`string`*

Adds the specified text string in the status section of the status bar (window right to the busy indicator). The text remains until it is overwritten.

This command is usually used to print transient messages (for example, Loading library *`name`*).

Note: This command is only available in the legacy GUI.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_suspend

```
gui_suspend
```

Suspends the automatic GUI updates.

Note: This command is only available in the legacy GUI.

Related Information

Related command:	gui_resume on page 222

gui_update

```
gui_update
```

Updates the GUI. This is the same as selecting *Update GUI* from the File menu. It forces synchronization of primary viewers with the current design database. You can also update the GUI by clicking 'Redraw' button on the GUI Viewer.

Related Information

Related commands:	gui_resume on page 222
	gui_suspend on page 224

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

HDL Viewer GUI Text Commands

- gui_hv_clear on page 226
- gui_hv_get_file on page 226
- gui_hv_load_file on page 226
- gui_hv_set_indicators on page 227

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_hv_clear

`gui_hv_clear`

Removes all the data in the HDL Viewer.

Note: This command is only available in the legacy GUI.

gui_hv_get_file

`gui_hv_get_file`

Returns the name of the file currently loaded in HDL Viewer.

Note: This command is only available in the legacy GUI.

gui_hv_load_file

`gui_hv_load_file` *`filename`*

Loads the specified file name into the HDL Viewer. Same as clicking the *Open HDL File* icon in the HDL Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`filename`	The name of the HDL file to load.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_hv_set_indicators

`gui_hv_set_indicators [-clear] line_number column_number`

Sets the *line* and *column* number indicators. Using this command is useful if you are writing a script and you want to highlight a specific line in the HDL Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-clear`	Removes all tag highlighting. If this option is not used, then the specified line number is highlighted.
column_number	Specifies the column number. If the *column_number* argument is not specified, then the default is `0`.
line_number	Specifies the line number.

Example

- The following command highlights the specified line number and sets the line number to `12` and the column number to `10`, as shown in Figure 5-2.

  ```
  legacy_genus:/> gui_hv_set_indicators 12 10
  ```

- The following commands sets the line number to 12, the column number to 10, and removes the highlighting:

  ```
  legacy_genus:/> gui_hv_set_indicators 12 10 -clear
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Figure 5-2 Setting Line and Column Numbers in the HDL Viewer

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Schematic Viewer GUI Text Commands

- gui_sv_clear on page 230
- gui_sv_cone on page 230
- gui_sv_get_instance on page 231
- gui_sv_grey on page 231
- gui_sv_highlight on page 232
- gui_sv_load on page 233
- gui_sv_snapshot on page 234
- gui_sv_toolbar_button on page 234

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_sv_clear

```
gui_sv_clear
```

Clears highlighting and selection lists in the Schematic Viewer. You can select the Schematic Viewer using + sign.

Related Information

Related command: gui_sv_highlight on page 232

gui_sv_cone

```
gui_sv_cone [-append] instance
```

Displays the specified instance into a new schematic cone viewer.

Note: This command is the text -equivalent for *Open in —Schematic Viewer (cone)* command in the Schematic Viewer, which is available when an instance is selected.

Options and Arguments

`-append`	Appends the new cone to the last open cone schematic.
`instance`	Specifies the instance you want to start drawing a cone from (the instance that you would have highlighted in the schematic before bringing up the context-sensitive menu).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_sv_get_instance

```
gui_sv_get_instance
```

Returns the object of design or hierarchical instance which is currently displayed on the Schematic Viewer.

Examples

- If top design is displayed on the Schematic Viewer, gui_sv_get_instance returns the design object:

  ```
  gui_sv_get_instance
  myAlu
  ```

- If a hierarchical instance is displayed on the Schematic Viewer, `gui_sv_get_instance` returns the instance object.

  ```
  gui_sv_get_instance
  myAlu/ckGen
  ```

gui_sv_grey

```
gui_sv_grey [on | off]
```

Controls the grey mode. If the grey mode is on, Schematic Viewer in displayed in monochrome. You can also right click in the Schematic Viewer and select *Grey Mode On* or *Grey Mode Off.*

Options and Arguments

`-on`	Turns on grey mode in the Schematic Viewer.
`-off`	Turns off grey mode in the Schematic Viewer.
	If you specify neither -`on` or -`off`, then the grey mode is turned `off` by default.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_sv_highlight

```
gui_sv_highlight [-append] [-color color] [-center]
     [-from {port|pin}] [-to {port|pin}] [-zoomto]
     {port|pin|net|instance}
```

Highlights the specified object in the Schematic Viewer.

Options and Arguments

`-append`	Highlights the specified object, retaining existing highlighted objects. If you omit this option, all existing highlighting is cleared and only the specified object in highlighted by default
`-center`	Centers an object in the Schematic Viewer.
`-color color`	Specifies the color for highlighting an object.
`-from {port|pin}`	Specifies the start point of the highlighted net.
`-to {port|pin}`	Specifies the end point of the highlighted net.
`-zoomto`	Zooms into the specified port, pin, instance or net.

Examples

- The following Tcl procedure highlights inverters in the Schematic Viewer

```
# highlight inverters
proc highlight_inverters {idir type odir} {
    # only proceed if no object under cursor
    if {$type != "none"} return
    # clear any highlight and selection
    gui_sv_clear
    _find_inverters $idir
}

proc _find_inverters {idir} {
    # search for inverters
    foreach inst [find $idir -maxdepth 2 -instance *comb/*] {
        if {[get_attribute inverter $inst] == "true"} {
            gui_sv_highlight $inst -append -color green
        }
    }
}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example highlights the falling clock edge flip-flops

```
# highlight falling clock edge flip-flops
proc highlight_falling_edge_ff {idir type odir} {
    # only proceed if no object under cursor
    if {$type != "none"} return
    # clear any highlight and selection
    gui_sv_clear
    _find_falling_edge_ff $idir
}

proc _find_falling_edge_ff {idir} {
    # search for sequential gates with falling clock edge
    foreach inst [find $idir -maxdepth 2 -instance *seq/*] {
        set libcell [get_attribute libcell $inst]
        if {$libcell == ""} continue
        if {[get_attribute flop $libcell] == "true"} {
            foreach libpin [find $libcell -libpin *] {
                if {[get_attribute output $libpin] == "true"} {
                    foreach arc [get_attribute incoming_timing_arcs $libpin] {
                        set liberty [get_attribute liberty_attributes $arc]
                        if {[lsearch $liberty falling_edge] != -1} {
                            gui_sv_highlight $inst -append
                        }
                    }
                }
            }


        }
    }
}
```

gui_sv_load

`gui_sv_load {`*`design`* `|` *`instance`*`}`

Specifies an hierarchical instance or instance or design to load into the main Schematic Viewer.

Note: The opened window should be schematic viewer in order for this command to work.If you are in any other viewer, open the schematic viewer before executing this command.

Examples

- Following example shows how to load an instance in Schematic Viewer.

```
gui_sv_load TDSP_CORE_INST
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_sv_snapshot

```
gui_sv_snapshot
     [-overwrite] [-no_fit] [-ps] file
```

Saves a snapshot image of the part of the design that is visible in the Schematic Viewer.

Options and Arguments

file	Specifies the name of the file in which to save the snapshot.
-no_fit	Captures the design as is exactly displayed on the Schematic Viewer (as is zoomed or highlighted). If you omit this option, whole design is fit to the full screen and saved (regardless of how it is displayed).
-overwrite	Specifies to overwrite an existing file. If you omit this option, you'll get a message that indicates that the file exists.
-ps	Specifies to create the file in PostScript format. If you omit this option, a GIF file will be created.

gui_sv_toolbar_button

```
gui_sv_toolbar_button
     -icon string -proc string -tip string
```

Adds a customized button to the toolbar of the schematic viewer.

Options and Arguments

-icon *string*	Specifies the GIF or PNM file that contains the drawing of the icon. The icon size is 24x24 and has a gray94 background.
-proc *string*	Specifies the name of the Tcl procedure to execute.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	The procedure must be loaded before you press the button to call it.
`-tip` *`string`*	Specifies the tool tip to be displayed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Physical Viewer GUI Text Commands

- add_gui_shape on page 238
- deselect_obj on page 240
- gui_pv_airline_add on page 241
- gui_pv_airline_add_custom on page 243
- gui_pv_airline_delete on page 244
- gui_pv_airline_display on page 245
- gui_pv_airline_raw_add on page 246
- gui_pv_airline_raw_add_custom on page 248
- gui_pv_align_instance_to_boundary on page 249
- gui_pv_align_instances on page 249
- gui_pv_clear on page 250
- gui_pv_connectivity_airlines on page 251
- gui_pv_deselect on page 252
- gui_pv_display_collection on page 253
- gui_pv_draw_box on page 254
- gui_pv_draw_circle on page 256
- gui_pv_draw_line on page 258
- gui_pv_draw_net on page 259
- gui_pv_draw_triangle on page 260
- gui_pv_get_design on page 261
- gui_pv_grey on page 262
- gui_pv_highlight on page 263
- gui_pv_highlight_hier_instances on page 265
- gui_pv_highlight_update on page 266
- gui_pv_label on page 267
- gui_pv_lineup_instances on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- gui_pv_new_viewer on page 268
- gui_pv_preferences on page 270
- gui_pv_redraw on page 271
- gui_pv_select on page 272
- gui_pv_selection on page 273
- gui_pv_snapshot on page 274
- gui_pv_steiner on page 277
- gui_pv_steiner_tree on page 278
- gui_pv_toolbar_button on page 279
- gui_pv_update on page 279
- gui_pv_zoom_box on page 280
- gui_pv_zoom_fit on page 280
- gui_pv_zoom_in on page 281
- gui_pv_zoom_out on page 281
- gui_pv_zoom_to on page 283
- select_obj on page 283

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_gui_shape

```
add_gui_shape
     [-layer
     {red|blue|green|yellow|magenta|cyan|lightpink|purple|teal|olive|plum|navy|pi
     nk|lime|orange|brown|lightblue|gold|chocolate|lightgreen|maroon|salmon|viole
     t|darkcyan|royalblue|darkgreen|tomato|chartreuse|wheat|deepskyblue|darkorang
     e|darkred|white}
     [-style {cross|outline|solid}]
     [-width integer] [-line float...]
     [-polygon float...] [-rect string]
```

Adds a specified shape to the Physical Viewer.

Options and Arguments

`-arrow` Specifies to draw arrow in middle of a line.

`-layer {red|blue|green|yellow|magenta|cyan|lightpink|purple|teal|olive|plum|navy|pink|lime|orange|brown|lightblue|gold|chocolate|lightgreen|maroon|salmon|violet|darkcyan|royalblue|darkgreen|tomato|chartreuse|wheat|deepskyblue|darkorange|darkred|white}`

Specifies the layer color.

Default: `red`

`-line` *`float`* Specifies the line segment coordinates in the format: `{x1 y1 x2 y2 ...}`

`-polygon` *`float`* Specifies polygon coordinates in the format: `{x1 y1 x2 y2 ...}`

`-rect` *`string`* Specifies rectangle coordinates in the format: `{x1 y1 x2 y2 ...}`

`-style {cross|outline|solid}`

Specifies the layer style.

Default: `solid`

`-width` *`integer`* Specifies the border width.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

deselect_obj

```
deselect_obj [-all] {fpobject | region | blockage | net | port | pin | instance
     }...
```

Deselects the specified objects in the Physical Viewer.

Options and Arguments

`-all`	Specifies to deselect all objects.
`{fpobject | region | blockage | net | port | pin | instance}`	Specifies the objects to deselect.

Related Information

Related command: select_obj on page 283

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_add

```
gui_pv_airline_add -name name [-id integer]
     -from {pin|port|instance} -to {pin|port|instance}
     [-color color] [-glyph {arrow|circle|none}]
     [-nodisplay] [-width integer]
```

Adds the specified airline to the Physical Viewer.

Options and Arguments

`-color color`
: Specifies the airline color.

  The color can have following values:

  `blue, brown, cyan, green, magenta, orange, pink, purple, red, yellow, white.`

  Default: `blue`

`-from {pin|port|instance}`
: Specifies the from object.

`-id integer`
: In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

  Note: This option is only available in the legacy GUI.

`-glyph {arrow | circle | none}`
: Specifies the glyph type.

  Note: This option is not available in the legacy GUI.

`-name name`
: Specifies the airline name.

`-nodisplay`
: Specifies not to display the airline.

`-to {pin|port|instance}`
: Draws an airline from the center of the from object to the center of the to object.

`-width integer`
: Specifies the airline width.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following command creates a `red` airline named `test` from the `g442` port to the `g412` port, as shown in Figure 5-3.

```
gui_pv_airline_add -from g442 -to g412 -color red -name test
```

Figure 5-3 Specified Airline in the Physical Viewer

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_add_custom

```
gui_pv_airline_add_custom -name name [-id integer]
     -from {port|instance} -to {pin|port|instance}
     [-color color] [-label label | -nolabel]
     [-nodisplay] [-noglyph] [-nomiddle]
     [-type {0|1|2}] [-width integer]
```

Adds a customized airline between two objects in the Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-color` *`color`*	Specifies the airline color. *Default*: `blue`
`-from {`*`pin`*`|`*`port`*`|`*`instance`*`}`	Specifies the from object.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`label`*	Specifies a label to place at the center of the airline.
`-name` *`name`*	Specifies the airline name.
`-nodisplay`	Specifies not to display the airline at the time you create it.
`-noglyph`	Specifies not to display the airline glyphs.
`-nolabel`	Specifies not to display the airline label.
`-nomiddle`	Specifies not to display the airline arrow in the middle.
`-to {`*`pin`*`|`*`port`*`|`*`instance`*`}`	Draws an airline from the center of the from object to the center of the to object.
`-type {0|1|2}`	Specifies the airline glyph type at the endpoints of the airline. 0—no glyphs are added 1—arrow type glyphs are added 2—circle type glyphs are added
`-width` *`integer`*	Specifies the airline width.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_airline_delete

`gui_pv_airline_delete name [-id integer]`

Deletes airlines from the Physical Viewer with the specified *name*.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
name	Specifies the airline name to be deleted. The airline name must have been added using the gui_pv_airline_add command.

Example

- The following command removes the airline `test` from the Physical Viewer:

```
legacy_genus:/> gui_pv_airline_delete test
```

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_display

`gui_pv_airline_display` *`name`* `[-id` *`integer`*`]`

Displays the specified airline name in the Physical Viewer.

Options and Arguments

`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`name`	Specifies the airline name to be displayed. The airline name must have been added using the gui_pv_airline_add command.

Example

- The following command displays the airline `test` in the Physical Viewer:

```
legacy_genus:/> gui_pv_airline_display test
```

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_raw_add

```
gui_pv_airline_raw_add [-name name] [-id integer]
     [-color color] [-label label] [-width integer]
     [-nodisplay] [-glyph {arrow | circle | none}]
     -fx float -fy float
     -tx float -ty float
```

Creates an airline in the Physical Viewer between two points specified by their coordinates.

Options and Arguments

`-color` *`color`*	Specifies the airline color. The color can have following values: `blue, brown, cyan, green, magenta, orange, pink, purple, red, yellow, white.` *Default*: `blue`
`-fx` *`float`*	Specifies the x coordinate of the from object.
`-fy` *`float`*	Specifies the y coordinate of the from object.
`-glyph {arrow | circle | none}`	Specifies the glyph type. **Note:** This option is not available in the legacy GUI.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`-label` *`label`*	Specifies a label to place at the center of the airline.
`-name` *`name`*	Specifies the airline name.
`-nodisplay`	Specifies not to display the airline.
`-tx` *`float`*	Specifies the x coordinate of the to object.
`-ty` *`float`*	Specifies the y coordinate of the to object.
`-width` *`integer`*	Specifies the width of the airline.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_airline_raw_add_custom

```
gui_pv_airline_raw_add_custom -name name [-id integer]
     -fx float -fy float -tx float -ty float
     [-color color] [-label label | -nolabel]
     [-nodisplay][-noglyph] [-nomiddle]
     [-type {0|1|2}] [-width integer]
```

Creates a customized airline in the Physical Viewer between two points specified by their coordinates.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-color` *`color`*	Specifies the airline color. *Default*: `blue`
`-fx` *`float`*	Specifies the X coordinate of the from object.
`-fy` *`float`*	Specifies the Y coordinate of the from object.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`label`*	Specifies a label to place at the center of the airline.
`-name` *`name`*	Specifies the airline name.
`-nodisplay`	Specifies not to display the airline at the time you create it.
`-noglyph`	Specifies not to display the airline glyphs.
`-nolabel`	Specifies not to display the airline label.
`-nomiddle`	Specifies not to display the airline arrow in the middle.
`-tx` *`float`*	Specifies the X coordinate of the to object.
`-ty` *`float`*	Specifies the Y coordinate of the to object.
`-type {0|1|2}`	Specifies the airline glyph type at the endpoints of the airline. 0—no glyphs are added 1—arrow type glyphs are added 2—circle type glyphs are added
`-width` *`integer`*	Specifies the airline width.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_align_instance_to_boundary

```
gui_pv_align_instance_to_boundary
     [-top] [-bottom] [-left] [-right] instance
```

Aligns the specified instance with the specified edge of the die.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-bottom`	Aligns the instance with the bottom edge of the die.
`instance`	Specifies the instance to be aligned.
`-left`	Aligns the instance with the left edge of the die.
`-right`	Aligns the instance with the right edge of the die.
`-top`	Aligns the instance with the top edge of the die.

gui_pv_align_instances

```
gui_pv_align_instances
     [-top] [-bottom] [-left] [-right] instance...
```

Aligns the specified instances with one another.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-bottom`	Aligns the bottom sides of the specified instances.
`instance`	Specifies the instances to be aligned.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-left`	Aligns the left sides of the specified instances.
`-right`	Aligns the right sides of the specified instances.
`-top`	Aligns the top sides of the specified instances.

gui_pv_clear

```
gui_pv_clear [-airline] [-highlight] [-selection]
     [-shape] [-all] [-id integer]
```

Clears airlines or highlighting from the Physical Viewer, or deselects objects in the Physical Viewer. When no options are specified, everything is cleared.

Options and Arguments

`-airline`	Clears all airlines.
`-all`	Clears in all Physical Viewers. **Note:** This option is only available in the legacy GUI.
`-highlight`	Clears all highlighting.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`-selection`	Deselects all objects.
`-shape`	Clears all shapes. **Note:** This option is not available in the legacy GUI.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_connectivity_airlines

```
gui_pv_connectivity_airlines [-id integer]
     [-append] {instance | port }...
```

Shows the connectivity airlines from the specified instance or port to the instances and ports it is connected to.

Options and Arguments

`-append`	Adds the new airlines to the existing ones.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`{`*`instance`* `|` *`port`*`}`	Specifies the name of the instance or port from where to draw the connectivity airlines.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_deselect

`gui_pv_deselect` *`object...`* `[-all] [-id` *`integer`*`]`

Specifies the list of object types to deselect in the physical viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-all`	Deselects everything from the Physical Viewer.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. The ID number of the physical viewer is displayed in the title bar.
`object`	Object to deselect. You can specify any of the following objects: `blackbox, blockage, bump, cluster, def_pin, gcell, instance, pcell, pdomain, pin, port, region, row`

Related Information

CCMPR02024153

Related commands: gui_pv_new_viewer on page 268
gui_pv_selection on page 273

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_display_collection

`gui_pv_display_collection [-group group] [-id integer]`

Displays a collection of objects in the Physical Viewer.

Options and Arguments

`-group group`	Specifies the name of the group to be displayed. A group can be created and appended to by any `gui_pv_xx` command that has the `-collect` and `-group` options.
`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

Related Information

Related commands:

gui_pv_draw_box on page 254
gui_pv_draw_circle on page 256
gui_pv_draw_line on page 258
gui_pv_draw_triangle on page 260
gui_pv_highlight on page 263
gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_draw_box

```
gui_pv_draw_box -llx float -lly float
     -width float -height float
     [-color color] [-skip]
     [-collect] [-append] [-stipple]
     [-group name] [-label label] [-data data]
     [-dbu] [-id integer]
```

Draws a box on top of the physical view. You can use this command to annotate information.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-append`	Appends the object to highlight.
`-collect`	Collects objects to highlight.
`-color` *`color`*	Specifies the highlight color. *Default*: `red`
`-data` *`data`*	Specifies the object data string.
`-dbu`	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
`-group` *`name`*	Displays only those objects in the specified group. These objects were previously marked for display through the `gui_pv_highlight -collect` command.
`-height` *`float`*	Specifies the height of the box in micrometer.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`string`*	Specifies the object label.
`-llx` *`float`*	Specifies the x coordinate of the origin of the box in micrometer.
`-lly` *`float`*	Specifies the y coordinate of the origin of the box in micrometer.
`-skip`	Skips the color name check.
`-stipple`	Specifies to use stipple fill pattern to highlight the object.
`-width` *`float`*	Specifies the width of the box micrometer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_draw_circle

```
gui_pv_draw_circle -cx float -cy float -radius float
     [-color color] [-collect] [-append] [-stipple]
     [-group name] [-label label] [-data data]
     [-dbu] [-id integer]
```

Draws a circle on top of the physical view. You can use this command to annotate information.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-append`	Appends the object to highlight.
`-collect`	Collects objects to highlight.
`-color` *`color`*	Specifies the highlight color. *Default*: `red`
`-cx` *`float`*	Specifies the x coordinate of the center of the circle in micrometer.
`-cy` *`float`*	Specifies the y coordinate of the center of the circle in micrometer.
`-data` *`data`*	Specifies the object data string.
`-dbu`	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
`-group` *`name`*	Displays only those objects in the specified group. These objects were previously marked for display through the `gui_pv_highlight -collect` command.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`label`*	Specifies the object label.
`-radius` *`float`*	Specifies the radius of the circle in micrometer.
`-stipple`	Specifies to use stipple fill pattern to highlight the objects.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_draw_line

```
gui_pv_draw_line -fx float -fy float -tx float -ty float
     [-color color] [-stipple] [-collect] [-append]
     [-group name] [-label label] [-data data]
     [-dbu] [-id integer]
```

Draws a line between the specified coordinates on top of the physical view. You can use this command to annotate information.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-append`	Appends the object to highlight.
`-collect`	Collects objects to highlight.
`-color` *`color`*	Specifies the highlight color. *Default*: `red`
`-data` *`data`*	Specifies the object data string.
`-dbu`	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
`-fx` *`float`*	Specifies the x coordinate of the first (or from) point in micrometer.
`-fy` *`float`*	Specifies the y coordinate of the first (or from) point in micrometer.
`-group` *`name`*	Displays only those objects in the specified group. These objects were previously marked for display through the `gui_pv_highlight -collect` command.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`label`*	Specifies a label to place at the center of the line.
`-stipple`	Specifies to use stipple fill pattern to highlight the objects.
`-tx` *`float`*	Specifies the x coordinate of the second (or to) point in micrometer.
`-ty` *`float`*	Specifies the y coordinate of the second (or to) point in micrometer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_draw_net

`gui_pv_draw_net [-delete] [hnet...]`

Adds or removes the route drawing for the specified net.

Note: The command only works for eGR (HUM) routed nets. If there are no eGR routes, nothing happens.

Note: This command is not available in the legacy GUI.

Options and Arguments

`-delete`	Removes the net from the route drawing.
`hnet`	Specifies the net(s) for which to add the route drawing.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_draw_triangle

```
gui_pv_draw_triangle -llx float -lly float
     -width float -height float
     [-color color] [-stipple] [-collect] [-append]
     [-group name] [-label label] [-data data]
     [-dbu] [-id integer]
```

Draws an isosceles triangle on top of the physical view with the specified width and height that originates in the specified lower left coordinates. You can use this to annotate information.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-append`	Appends the object to highlight.
`-collect`	Collects objects to highlight.
`-color` *`color`*	Specifies the highlight color. *Default*: `red`
`-data` *`data`*	Specifies the object data string.
`-dbu`	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
`-group` *`name`*	Displays only those objects in the specified group. These objects were previously marked for display through the `gui_pv_highlight -collect` command.
`-height` *`float`*	Specifies the height of the triangle in micrometer.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-label` *`string`*	Specifies a label to place at the center of the triangle.
`-llx` *`float`*	Specifies the x coordinate of the lower left point in micrometer.
`-lly` *`float`*	Specifies the y coordinate of the lower left point in micrometer.
`-stipple`	Specifies to use stipple fill pattern to highlight the objects.
`-width` *`float`*	Specifies the width of the triangle in micrometer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_get_design

`gui_pv_get_design [-id integer]`

Returns the current design displayed in the specified Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_grey

`gui_pv_grey [on | off] [-id integer]`

Controls the grey mode. You can also right-click the mouse button in the Physical Viewer and select *Grey Mode On* or *Grey Mode Off.*

Note: This command is only available in the legacy GUI.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-off`	Turns off grey mode in the Physical Viewer.
`-on`	Turns on grey mode in the Physical Viewer.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_highlight

```
gui_pv_highlight [-color color] [-collect] [-append]
     [-group name][-pattern {cross|none|solid}] [-zoom]
     {blockage | extraction_grid | gcell | region | row
     | instance | pin | port | net}...
     [-data data] [-label string]
     [-stipple] [-instances] [-all] [-id integer]
```

Highlights objects in the Physical Viewer.

Options and Arguments

`-all`	Highlights in all Physical Viewers. **Note:** This option is only available in the legacy GUI.
`-append`	Appends the object to highlight.
{*blockage* | *extraction_grid* | *gcell* | *instance* | *net* | *pin* | *port* | *region* | *row*}	Specifies the object to highlight if it is in the scope of the current Physical Viewer.
`-collect`	Collects objects to highlight
`-color` *color*	Specifies the color for highlighting. Possible values: {red|blue|green|yellow|magenta|cyan|lightpink|purple|teal|olive|plum|navy|pink|lime|orange|brown|lightblue|gold|chocolate|lightgreen|maroon|salmon|violet|darkcyan|royalblue|darkgreen|tomato|chartreuse|wheat|deepskyblue|darkorange|darkred|white} *Default*: `red`
`-data` *data*	Specifies the object data string. **Note:** This option is only available in the legacy GUI.
`-group` *name*	Specifies the group name for the object.
`-id` *integer*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`-instances`	Collects region instances.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Note: This option is only available in the legacy GUI.
`-label` *`string`*	Specifies the object label.
	Note: This option is only available in the legacy GUI.
`-pattern {cross | none | solid}`	
	Specifies the stipple pattern.
	Default: `none`
	Note: This option is not available in the legacy GUI.
`-stipple`	Specifies to use stipple fill pattern to highlight the object.
	Note: This option is only available in the legacy GUI.
`-zoom`	Zooms the highlighted objects.
	Note: This option is not available in the legacy GUI.

Examples

- The following command highlights the g411 port in yellow (see Figure 5-4 on page 264):

```
legacy_genus:/> gui_pv_highlight -color yellow g411
```

- The following command adds the g405 port to the highlighted list (see Figure 5-4 on page 264):

```
legacy_genus:/> gui_pv_highlight -color yellow -append g405
```

Figure 5-4 Highlighting an Object in the Physical Viewer

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command specifies the instance `eve1` to be highlighted with the `gui_pv_draw_collection` command. The `-group` option indicates that to which group eve1 should belong. In this case, it is `Laurence`:

```
legacy_genus:/> gui_pv_highlight -collect /designs/bree/instances_hier/eve1/ \
    -group laurence
```

Related Information

Related commands:	gui_pv_clear on page 250
	gui_pv_new_viewer on page 268

gui_pv_highlight_hier_instances

```
gui_pv_highlight_hier_instance [-threshold integer]
     [-depth integer] [-under instance]
     [-instances instance_list] [-fromgui] [design]
```

Highlights instances in the specified hierarchical instance in the Physical Viewer.

Options and Arguments

`-depth` *`integer`*	Specifies the search depth.
`design`	Specifies the design containing the instances to be highlighted.
`-fromgui`	Indicates whether the call comes from the GUI. **Note:** This option is not available in the legacy GUI.
`-instances` *`instance_list`*	Specifies the instances to be highlighted.
`-threshold` *`integer`*	Specifies the threshold of the hierarchical instance count.
`-under` *`instance`*	Specifies the parent hierarchical instance that contains the instances to be highlighted. **Note:** Options `-under` and `-instances` are mutually exclusive.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_highlight_update

```
gui_pv_highlight_update -property string [-id integer]
     -value string [-group string]
     {blockage | bump | gcell | instance | pcell | port | region}...
```

Updates the object highlight in the Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`{blockage|bump	gcell|instance|pcell|port|region}`	Specifies the names of the objects for which to update the highlight.
`-group string`	Specifies the object group name.
`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-property value`	Specifies the property to update. You can update the colors, the labels, or the stipple pattern.
`-value value`	Specifies the object property value.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_label

```
gui_pv_label
     -x float -y float label
     [-color color] [-id integer]
```

Adds a text label at the specified point in the Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-color` *`color`*	Specifies the label color. *Default*: `white`.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`label`	Specifies the label name.
`-x` *`float`*	Specifies the x coordinate for the label.
`-y` *`float`*	Specifies the y coordinate for the label.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_lineup_instances

```
gui_pv_lineup_instances -space float
     [-top] [-bottom] [-left] [-right]
      [instance]...
```

Modifies the instance placement by evenly distributing the instances vertically or horizontally starting from the specified side at regular intervals.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-bottom`	Lines up the instances starting from the bottom. The bottommost instance does not move.
`instance`	Specifies the instances to be distributed.
`-left`	Lines up the instances starting from the left. The leftmost instance does not move.
`-right`	Lines up the instances starting from the right. The rightmost instance does not move.
`-space` *`float`*	Specifies the interval spacing in micron.
`-top`	Lines up the instances starting from the top. The topmost instance does not move.

gui_pv_new_viewer

```
gui_pv_new_viewer [design]...
```

Opens a new Physical Viewer.

Note: The id number of the physical viewer is displayed in the title bar.

Options and Arguments

`design`	In case of multiple designs, specifies the design to display in the new Physical Viewer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This argument is optional if there is only one design in memory.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_preferences

```
gui_pv_preferences -name {true | false | 1 | 0}
     -selectable {true | false | 1 | 0}
     -visible {true | false | 1 | 0}
     [-color string]  -style {outline | fill | stipple}
     string [-id integer]
```

Configures the preferences for the specified object type in the Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-color` *`string`*
: Specifies the color of the objects of the specified type.

`-id` *`integer`*
: In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

`-name {true | false | 1 | 0}`
: Specifies to display the name of the objects of the specified type.

`-selectable {true | false | 1 | 0}`
: Specifies that objects of the specified type are selectable.

`string`
: Specifies the object type for which to configure the preferences. You can specify any of the following: `blackbox`, `blockage_placement`, `blockage_derived_placement`, `blockage_partial_placment`, `blockage_soft_placement`, `blockage_routing`, `bump`, `cluster`, `core`, `die`, `fence`, `gcell`, `guide`, `macro_cover`, `macro_fixed`, `macro_generic`, `macro_placed`, `pcell`, `pin`, `port`, `power_domain`, `region`, and `row`.

`-style {outline | fill | stipple}`
: Specifies the drawing style for the object type.

  Default: `outline`.

`-visible {true | false | 1 | 0}`
: Specifies that the objects of the specified type must be visible.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_redraw

`gui_pv_redraw [-id integer]`

Redraws the contents of the Physical Viewer.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_select

`gui_pv_select` *`object...`* `[-id` *`integer`*`]`

Specifies the list of object types to select in the Physical Viewer.

You can specify any of the following:

`blackbox`	`blockage`	`bump`
`cluster`	`def_pin`	`gcell`
`instance`	`pcell`	`pdomain`
`pin`	`pinstance`	`port`
`region`	`row`	

Note: This command is only available in the legacy GUI.

Options and Arguments

`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_selection

`gui_pv_selection [-id integer]`

Returns the list of selected objects in the Physical Viewer.

Note: You can select multiple objects by pressing the Shift key while drawing a region with the left mouse button. All objects overlapping that region are selected.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_snapshot

```
gui_pv_snapshot
     [-congestion] [-pin_density] [-utilization]
     [-no_fit] [-overwrite] [-png] [-ps] file [-id integer]
```

Saves a snapshot of the part of the design that is visible in the Physical Viewer.

Options and Arguments

`-congestion`	Overlays a congestion map on top of what is visible in the Physical Viewer before making a snapshot. **Note:** This option eliminates the need to first turn on the display of the congestion map.
`file`	Specifies the name of the file in which to save the snapshot.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`-no_fit`	Specifies to not fit the snapshot to the full screen. **Note:** This option is only available in the legacy GUI.
`-overwrite`	Specifies to overwrite an existing file. If you omit this option, you'll get a message that indicates that the file exists.
`-pin_density`	Overlays the pin density map on top of what is visible in the Physical Viewer before making a snapshot. **Note:** This option is only available in the legacy GUI.
`-png`	Specifies to create the file in PNG format. If this option is omitted, a GIF file will be created. **Note:** This option is not available in the legacy GUI.
`-ps`	Specifies to create the file in PostScript format. If this option is omitted, a GIF file will be created. **Note:** This option is only available in the legacy GUI.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-utilization`

Overlays a utilization map on the part of the design that is visible in the Physical Viewer before making a snapshot.

Note: This option eliminates the need to first turn on the display of the utilization map.

Note: This option is only available in the legacy GUI.

Examples

- The following command saves a snapshot of the utilization map in the `util.gif` file.

```
gui_pv_snapshot -utilization util.gif
```

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command saves a snapshot of the congestion map in the file with basename congest. Since the -ps option is not specified, the file is specified in GIF format.

```
gui_pv_snapshot -congestion congest
```

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_steiner

```
gui_pv_steiner
     [-icolor string] [-ocolor string]
     {instance | pin | port }... [-id integer]
```

Shows the Steiner routing from the specified object. For output pins show the route. For input pins the route for the driver is shown. For instances both routes for output pins and routes for the driver of input pins are shown.

Options and Arguments

`-icolor` *`string`*	Species the input pin color for the layer. *Default*: `green`.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`-ocolor` *`string`*	Species the output pin color for the layer. *Default*: `yellow`.
`{`*`instance`*`|`*`pin`*`|`*`port`*`}`	Specifies the name of the instance, pin, or port from where to draw the Steiner routing.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_steiner_tree

```
gui_pv_steiner_tree
     [-icolor string] [-ocolor string]
     {instance | pin | port }... [-id integer]
```

Shows the Steiner tree from the output pins of the specified instance or from the specified port.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-icolor` *`string`*	Species the input pin color for the layer. *Default*: `green`.
`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
`-ocolor` *`string`*	Species the output pin color for the instance.
`{`*`instance`* `|` *`pin`* `|` *`port`*`}`	Specifies the name of the instance or port from where to draw the Steiner trees.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_toolbar_button

```
gui_pv_toolbar_button
     -icon string -proc string -tip string [-id integer]
```

Adds a customized button to the toolbar of the Physical Viewer.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-icon` *`string`*	Specifies the GIF or PNM file that contains the drawing of the icon. The icon size is 24x24 and has a gray94 background.
`-proc` *`string`*	Specifies the name of the Tcl procedure to execute. The procedure must be loaded before you press the button to call it.
`-tip` *`string`*	Specifies the tool tip to be displayed.

gui_pv_update

```
gui_pv_update [string] [-design design]
```

Specifies the objects to be updated in the Physical Viewer. The object can be module, macro, instance or pin.

Note: This command is only available in the legacy GUI.

Options and Arguments

`-design` *`design`*	Specifies the design for which to update the GUI.
`string`	Specifies the objects to be updated in the Physical Viewer. You can specify row or region. By default, all objects will be updated.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_zoom_box

`gui_pv_zoom_box` *`llx lly urx ury`* `[-id` *`integer`*`]`

Zooms to the specified box in the Physical Viewer.

Options and Arguments

`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.
`llx lly urx ury`	Specifies the lower left and upper right coordinates of the box to zoom in to. You can specify floating numbers. The coordinates are specified in user units.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_zoom_fit

`gui_pv_zoom_fit [-id` *`integer`*`]`

Performs a “zoom fit” command in the Physical Viewer.

Options and Arguments

`-id` *`integer`*	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_zoom_in

`gui_pv_zoom_in [-id integer]`

Performs a “zoom in” command in the Physical Viewer.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

Related Information

Related command: gui_pv_new_viewer on page 268

gui_pv_zoom_out

`gui_pv_zoom_out [-id integer]`

Performs a “zoom out” command in the Physical Viewer.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: gui_pv_new_viewer on page 268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

gui_pv_zoom_to

`gui_pv_zoom_to [-id integer]`

Zooms to the bounding box around selected objects in the Physical Viewer. Clicking 'Zoom Selected' button on the Physical Viewer also performs this command.

Options and Arguments

`-id integer`	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command. **Note:** This option is only available in the legacy GUI.

Related Information

Related command: gui_pv_new_viewer on page 268

select_obj

`select_obj {fpobject | region | blockage | port | pin | hpin | hinst |inst}...`

Selects the specified objects in the Physical Viewer.

Options and Arguments

`{fpobject|region|blockage|port|pin|hpin|hinst|nst}`

Specifies the objects to select.

Related Information

Related command: deselect_obj on page 240

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

6

Chipware Developer

- cwd on page 286
- cwd check on page 287
- cwd create_check on page 291
- cwd report_check on page 294
- hdl_create on page 297
- hdl_create binding on page 299
- hdl_create component on page 301
- hdl_create implementation on page 303
- hdl_create library on page 305
- hdl_create operator on page 306
- hdl_create package on page 307
- hdl_create parameter on page 309
- hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cwd

```
cwd {check | create_check | report_check}
```

Controls the ChipWare Developer (CWD) Linter in the ChipWare developer framework.

Options and Arguments

`check`	Invokes the CWD Linter.
`create_check`	Registers a check to the Linter.
`report_check`	Reports information about various check names and check points.

Related Information

Related commands:	cwd check on page 287
	cwd create_check on page 291
	cwd report_check on page 294

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cwd check

```
cwd check [-effort string]
     [-skip
       [ hdl_lib | hdl_comp | hdl_pack | hdl_oper
       | hdl_arch | hdl_impl | hdl_bind | hdl_param
       | hdl_pin]... ]
     [ [-summary [-verbose] | -quiet ]
       [ hdl_lib | hdl_comp | hdl_pack | hdl_oper
       | hdl_arch | hdl_impl | hdl_bind | hdl_param
       | hdl_pin]...]
     [> file]
```

Exercises checking rules and summarizes the outcome in various degrees of verboseness. The `cwd check` command can run on one or more of any of the following `hdl_objects`:

- `hdl_arch`
- `hdl_bind`
- `hdl_comp`
- `hdl_impl`
- `hdl_lib`
- `hdl_oper`
- `hdl_pack`
- `hdl_param`
- `hdl_pin`

There are various "check points" associated with these objects. A check point is a specific point in the flow in which the CWD objects are processed. There are various checking rules associated with CWD check points.

The Genus path to the `hdl_objects` to be checked can either be an absolute path:

```
legacy_genus:/> cwd check /hdl_libraries/CW/components/CW_add
```

Or it can be a relative path with respect to the current working directory:

```
legacy_genus:/> cd /hdl_libraries/CW/components
legacy_genus:/> cwd check CW_add
```

You can use wild cards for specifying multiple `hdl_objects`:

```
legacy_genus:/> cwd check /hdl_libraries/CW/components/*add*
```

or:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> cd /hdl_libraries/CW/components
legacy_genus:/> cwd check *add*
```

You can specify multiple directories at the same time:

```
legacy_genus:/> cwd check {/hdl_libraries/CW /hdl_libraries/DW02}
```

There are various “check points” associated with these objects. A check point is a specific point in the flow in which the CWD objects are processed. There are various checking rules associated with CWD check points.

By default, `cwd check` checks all `hdl_objects` underneath the specified set of `hdl_objects`. That is, it traverses the directory tree hierarchically and exercises all checks of all `hdl_objects` it traverses.

Options and Arguments

`-effort` *`string`*	Specifies the effort level. There are two effort levels: `low` and `medium`. The default effort level is `low`. `Low` — CWD linter runs checking rules that are registered as `low` effort. That is, it runs those checking rules that do not require reading any HDL code (of synthesis models). `Medium` — CWD linter runs checking rules that are registered as `low` and `medium` effort. That is, it runs those checking rules that may require parsing HDL code (of synthesis models) but do not require elaborating it. With each `medium` effort level check, the CWD linter automatically loads the HDL code before performing the check. For example, a check at this effort level may look at the `hdl_arch` of an `hdl_impl` and examine ordering of pins and parameters.
`file`	Specifies the filename to store the output of the command.
`hdl_lib` | *`hdl_comp`* | *`hdl_pack`* | *`hdl_oper`* | *`hdl_arch`* | *`hdl_impl`* | *`hdl_bind`* | *`hdl_param`* | *`hdl_pin`*	Specifies the `hdl_objects` to check.
`-quiet`	Only reports error and warning messages, if any. This is the recommended verbosity level when CWD linting is part of a routine process without any error expectations.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-skip {hdl_lib | hdl_comp | hdl_pack | hdl_oper | hdl_arch | hdl_impl | hdl_bind | hdl_param | hdl_pin}`

Specify one or more `hdl_objects` to skip.

`-summary`
: First reports error or warning messages, if any, and then produces a summary table of the pass/fail count of each checking rules exercised. This level of detail is the default verbosity level.

`-verbose`
: Produces a detailed report. In addition to the information produced by the `-summary` option, it also reports the pass/fail status of each check process exercised on each `hdl_object`.

Examples

- The following example runs checking rules on the `CW` libraries as well as all the other `hdl_objects` under it. The CWD linter will run all default (low-effort) mode checking rules up to the severity specified by the `information_level` attribute. A summary will be produced at the end.

```
legacy_genus:/> get_attribute information_level
1
legacy_genus:/> cwd check /hdl_libraries/CW
Check_name                              Passed Failed
-----------------------------------------------------
component_location                        146     0
component_sim_model_location              128     0
component_syn_model_is_vhdl               146     0
implementation_legality_formula           147     0
implementation_location                   147     0
implementation_preelab_script_location    147     0
non_builtin_implementation_location       147     0
package_default_location                    1     0
package_default_location_filesize           1     0
parameter_formula                         472     0
pin_bit_width                            1136     0
pin_parameter_in_bit_width               1114     0
-----------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example runs checking rules on all the `hdl_objects` under `parameters` and produces a verbose report:

```
legacy_genus:/> cwd check /hdl_libraries/CW/components/CW_mult/parameters/*
-verbose

checking param wA

Check ::cwd::parameter_formula::check_proc passed on /hdl_libraries/CW/
components/CW_mult/parameters/wA

checking param wB

Check ::cwd::parameter_formula::check_proc passed on /hdl_libraries/CW/
components/CW_mult/parameters/wB

    Check_name    Passed Failed
--------------------------------
parameter_formula    2      0
--------------------------------
```

- The following example will check the `CW_add` component, but will skip checking its bindings and implementations:

```
legacy_genus:/> cd /hdl_libraries/CW/components/CW_add

legacy_genus:/> cwd check . -skip { /hdl_libraries/CW/components/CW_add/
bindings/* \

    /hdl_libraries/CW/components/CW_add/implementations/* }
```

Related Information

Checking Rules in *Genus ChipWare Developer Guide for Legacy UI*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cwd create_check

```
cwd create_check
     -check_name string
     -severity integer
     -description string
     -checklist string
     [-effort {low|high}] [-force] [> file]
```

Registers user-defined checking rules for the CWD linter.

Registered rules can be reported using the `cwd report_check` command.

Options and Arguments

`-check_name` *`string`*
: Specifies an unique name for a new checking rule.

`-checklist` *`string`*
: Specifies, as a Tcl list of Tcl lists, checkpoint and Tcl proc pairs. The specification therefore would be in the form:

  `-checklist {{point_1 proc_1} {point_2 proc_2}}`

  Every sub-list has two elements. The first element is the name of a check point, defined by Genus. The second element is the name of a Tcl proc, defined by the user.

  Each sub-list specifies a check proc that is to be called at a certain check point. This check proc will be executed every time the flow reaches this check point.

  This Tcl list has one or more sub-lists. One checking rule can be associated with one or more check points.

  The check procs may or may not be allowed to parse or elaborate the HDL code of the synthesis model, depending on the effort level of this checking rule. The check procs may print out error, warning, or info messages. Each check proc should return a string whose value is either `PASS` or `FAIL`.

`-description` *`string`*
: Specifies a character string that concisely describes what this checking rule examines.

`-effort {low|medium}`
: Specifies the effort level.

  Default: `low`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	`low` — The check is not allowed to parse the HDL code of the synthesis models. It can check correctness, consistency, or completeness of the CWD registration, including availability of UNIX files referred to by the `location` and `sim_model` attributes. `medium` — The check can read, load, and parse the HDL code of the synthesis models, but it cannot elaborate or synthesize the HDL code. When exercising such a checking rule, the HDL code is automatically loaded before checking is performed.
file	Specifies the filename to store the output of the command.
`-force`	Removes the existing checkname and adds the current specification. Alternatively, the same checkname will now correspond to the new definition.
`-severity` *integer*	Specifies the severity level of the message produced by this checking rule. The possible values (`0` through `10`) are the same as those for the `information_level` attribute. `-severity 0`: It is an error message to violate this rule `-severity 1`: It is a warning message to violate this rule `-severity 2`: It is a level 2 info message to violate this rule Severity levels 3 through 10 are all info messages at their respective levels.

Example

The following example defines the name of the check to be `arch_pin_order`. It is a medium effort level check: it has to be performed after the HDL code has been loaded. The severity of the check is 0, which means it is an error if this check fails. This checking rule is associated with two checkpoints, `HDL_ARCH_PINS_SCANNED` and `HDL_COMP_PINS_SCANNED`. At the `HDL_ARCH_PINS_SCANNED` checkpoint, a Tcl proc named `check_arch_pin_order` is to be called to perform this check. At the `HDL_COMP_PINS_SCANNED` checkpoint, a Tcl proc named `check_component_pin_order` is to be called to perform this check.

```
legacy_genus:/> cwd create_check -check_name "arch_pin_order" -effort "medium" \
    -severity 0 -description "Check Whether the order of pins specified \
    in the synthesis model is consistent with what is defined in the
    registration script"
    -checklist { {HDL_ARCH_PINS_SCANNED check_arch_pin_order} \
    {HDL_COMP_PINS_SCANNED check_component_pin_order} }
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Checking Rules in *Genus ChipWare Developer Guide for Legacy UI*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cwd report_check

```
cwd report_check
     [-checkpoint string] [-checkname string]
     [-max_width string] [> file]
```

Reports the registered checking rules. With each checking rule, it lists the:

- Name of the checking rule
- Checkpoint(s) the rule is associated with
- Check proc(s) attached to the associated checkpoint(s)
- Effort level of the rule
- Severity level of the rule
- Description string of the rule

Note: The `-checkname` and `-checkpoint` options cannot be used simultaneously.

Options and Arguments

`-checkname` *`string`*	Specifies, by name, a set of checking rules to report. Asterisk(*) can be used to see all names for `check_name`.
`-checkpoint` *`string`*	This switch specifies a set of checkpoints to report. Asterisk(*) can be used to see all names for checkpoint.
`-max_width` *`string`*	Limits the width of the columns in the table produced by this command. Limiting the width of a column to zero means removing that column from the table. This option takes a Tcl list of Tcl lists. Each sub-list represents a column in the table produced by this command. Each sub-list should have two elements: the first being name of the column (as seen in the report) and the second being an integer representing the maximum number of characters allowed for this column in the table. Default value is `{{check_name 40} {checkpoint 30} {check_proc 30} {effort 6}{severity 8} {description 30}}`
`file`	Specifies the filename to store the output of the command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example reports all checking rules that have been registered:

```
legacy_genus:/> cwd report_check -checkname {*}
```

- This reports info about all checkpoints:

```
legacy_genus:/> cwd report_check -checkpoint {*}
```

- The following example reports details about the checking rule `arch_pin_order`, which uses two check procs to associate with two checkpoints.

```
legacy_genus:/> cwd report_check -checkname {arch_pin_order} -max_width \
    {{Check_name 14} {Checkpoint 12} {Check_proc 15} {Effort 3} \
    {Severity 4} {Description 20}}
This example reports details about the checking rule arch_pin_order, which uses
two check procs to associate with two checkpoints.
-----------------------------------------------------------------------
Check_name    Checkpoint     Check_proc   Eff Seve      Description
                                          ort rity
-----------------------------------------------------------------------
arch_pin_order HDL_ARCH_PIN ::cwd::arch_pin med   0  Check whether the
               S_SCANNED    _order::cwd_pro ium      order of pins
                            c                        specified in the
                                                     synthesis model is
               HDL_COMP_PIN ::cwd::componen          consistent with
               S_SCANNED    t_pin_order::ch          what is defined in
                            eck_proc                 the registration
                                                     script
-----------------------------------------------------------------------
```

- The following example reports details about the checkpoint `HDL_OPER_BINDINGS_SCANNED`:

```
legacy_genus:/> cwd report_check -checkpoint {HDL_OPER_BINDINGS_SCANNED}
    -max_width {{Check_name 10} {Checkpoint 15} {Check_proc 15}
    {Effort 3} {Severity 4} {Description 20}}
-----------------------------------------------------------------------
Check_name    Checkpoint      Check_proc   Eff Seve      Description
                                           ort rity
-----------------------------------------------------------------------
operator_b HDL_OPER_BINDIN ::cwd::operator low   1  check that for
indings    GS_SCANNED      _bindings::chec          every hdl_bindings
                           k_proc                   defined for the
                                                    hdl_operator there
                                                    is at least one
                                                    attribute is set to
                                                    false
-----------------------------------------------------------------------
```

- The following example reports all checking rules whose checkname contains string "pin":

```
legacy_genus:/> cwd report_check -checkname {*pin*}
```

- The following example reports information about the `arch_pin_order` and `arch_parameter_order` checking rules:

```
legacy_genus:/> cwd report_check -checkname {arch_pin_order
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
arch_parameter_order}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create

```
hdl_create { binding | component | implementation | library
           | operator | package | parameter | pin}
```

Creates an HDL object for ChipWare developer.

Options and Arguments

`binding`	Creates a binding between a synthetic operator and a ChipWare component.
`component`	Creates a ChipWare component.
`implementation`	Creates a synthesis model for a ChipWare component.
`library`	Creates a synthetic library to hold ChipWare components, bindings, and implementations.
`operator`	Creates a synthetic operator.
`package`	Creates a package in the Design Information Hierarchy to hold the contents of a VHDL package.
`parameter`	Creates a parameter for a synthetic ChipWare component
`pin`	Creates an input/output/inout pin for a synthetic operator or component

Related Information

Related commands:

hdl_create binding on page 299
hdl_create component on page 301
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create operator on page 306
hdl_create package on page 307
hdl_create parameter on page 309

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create_pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create binding

```
hdl_create binding binding_name
     -operator operator_name
     [hdl_comp | bindings]
```

Creates a binding between a synthetic operator and a synthetic module. A synthetic module is also known as a ChipWare component.

Tip

Run-time can be reduced if you change directory (`vcd`) to the to the *`component name`* directory before issuing the command instead of specifying the entire component path name.

Options and Arguments

`binding_name`	Specifies the name of the binding that will be created.
*`hdl_comp	bindings`*	Specifies the path name of the component that holds this binding.
`-operator`	Specifies the synthetic operator that will be bound by this binding.

Examples

- The following examples both create a binding named `test1` for the `MY_MULT_OP` operator. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command.

```
legacy_genus:/hdl_libraries/my_CW/components/my_CW_mult> hdl_create binding \
    test1 -operator MY_MULT_OP
legacy_genus:/hdl_libraries/my_CW/components/my_CW_mult/bindings> ls
./      test1
```

- This example also creates a binding named `test1`. However, the command is issued from the root directory and therefore assumes a run-time penalty.

```
legacy_genus:/> hdl_create binding test1 -operator MY_MULT_OP \
    /hdl_libraries/my_CW/components/my_CW_mult
legacy_genus:/> ls /hdl_libraries/my_CW/components/my_CW_mult/bindings
./      test1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

CWD Component in *Genus ChipWare Developer Guide for Legacy UI*

Related commands:

hdl_create component on page 301
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create operator on page 306
hdl_create package on page 307
hdl_create parameter on page 309
hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create component

```
hdl_create component component_name
     [hdl_lib | components]
```

Creates a ChipWare component.

Options and Arguments

component_name Specifies the name of the component that will be created.

hdl_lib | components

Specifies the path name of the library that holds this component.

Examples

- The following examples both create a component named `CW_sweet_div`. However, the first example will save run-time over the second by `cd`'ing into the `components` directory and issuing the command:

```
legacy_genus:/hdl_libraries/CW/components> hdl_create component CW_sweet_div
legacy_genus:/hdl_libraries/CW/components> ls
...
CW_sweet_div
...
```

- This example also creates a component named `CW_sweet_div`. However, the command is issued from the root directory and therefore assumes a run-time penalty:

```
legacy_genus:/> hdl_create component CW_sweet_div /hdl_libraries/CW/
legacy_genus:/> ls /hdl_libraries/CW/components
...
CW_sweet_div
...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in the *Genus ChipWare Developer Guide for Legacy UI*

- CWD Component
- ChipWare Registration

Related commands:

hdl_create binding on page 299
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create operator on page 306
hdl_create package on page 307
hdl_create parameter on page 309
hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create implementation

```
hdl_create implementation implementation_name
     [-v1995 | -v2001 | -vhdl87 | -vhdl93] [-config string]
     [hdl_comp | implementations]
```

Creates an implementation for a ChipWare component. All components can have one or more implementations (or architectures) associated with them. This applies to both user and tool defended components. All implementations created with this command have a default priority of 1. The language option needs to be specified.

Options and Arguments

`-config string`
: Specifies the name of the configuration to be used.

  Use this option when the VHDL entity was specified with a configuration to create the component.

implementation_name
: Specifies the name of the implementation that will be created.

hdl_comp | implementation
: Specifies the path name of the component that owns this implementation.

`[-v1995 | -sv | -v2001 | -vhdl87 | -vhdl93]`
: Specifies the language version for the RTL code.

Example

- Both of the following examples create the `krystal` implementation in VHDL 1993 format for the `CW_sweet_div` component. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command:

```
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div> hdl_create
implementation \
    krystal -vhdl93
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div/implementations> ls
./      krystal
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- This example also creates an implementation named `krystal` for the same component. However, the command is issued from the root directory and therefore assumes a run-time penalty:

```
legacy_genus:/> hdl_create implementation krystal -vhdl93 \
    /hdl_libraries/CW/components/CW_sweet_div
legacy_genus:/> ls /hdl_libraries/CW/components/CW_sweet_div/implementations/
./      krystal
```

Related Information

See the following sections in the *Genus ChipWare Developer Guide for Legacy UI*

- CWD Component
- CWD Implementation
- Chipware Installation and Registration

Affects this attribute:	priority
Related commands:	hdl_create binding on page 299
	hdl_create component on page 301
	hdl_create library on page 305
	hdl_create operator on page 306
	hdl_create package on page 307
	hdl_create parameter on page 309
	hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create library

`hdl_create library` *`library_name`*

Creates a user defined HDL library. An HDL library can be a library of ChipWare components, a library of synthetic operators, or a VHDL library.

Options and Arguments

`library_name`	Specifies the name of the library that will be created.

Related Information

See the following sections in *Genus ChipWare Developer Guide for Legacy UI*

- CWD Component
- HDL Library
- ChipWare Installation and Registration

Related commands:

hdl_create binding on page 299
hdl_create component on page 301
hdl_create implementation on page 303
hdl_create operator on page 306
hdl_create package on page 307
hdl_create parameter on page 309
hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create operator

```
hdl_create operator operator_name
     [-signed | -unsigned]
```

Creates a synthetic operator. The operator type is unsigned. Genus has a set of tool-defined synthetic operators, but also provides the capability to add user-defined synthetic operators. All synthetic operators, whether they are tool or user-defined, are located under the operators directory(/hdl_libraries/synthetic/operators/).

Options and Arguments

operator_name	Specifies the name of the synthetic operator that will be created.
-signed	Specifies the created operator to be a signed operator.
-unsigned	Specifies the created operator to be an unsigned operator.

Related Information

Synthetic Operator in *Genus ChipWare Developer Guide for Legacy UI*

Related commands:

hdl_create binding on page 299
hdl_create component on page 301
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create package on page 307
hdl_create parameter on page 309
hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create package

```
hdl_create package pkg_name
     -path path_to_pkg
     [hdl_lib | packages]
```

Registers a VHDL package in the ChipWare Developer framework. Packages that are not registered are deleted after elaboration. However, registered packages are never deleted and their information can be further considered during synthesis as opposed to just during elaboration.

Registered packages are in the same location within Genus as non-registered packages:

`/hdl_libraries/library_name/packages/`

Options and Arguments

hdl_lib | *packages*	Specifies the path name of the library that holds this package.
-path *path_to_pkg*	Specifies the UNIX path name of the package to register.
pkg_name	Specifies the name of the package that will be created.

Examples

- Both of the following examples create the `numeric_std` package for the `ieee` library. However, the first example will save run-time over the second by `cd`'ing into the library name directory and issuing the command:

```
legacy_genus:/hdl_libraries/ieee/packages> hdl_create package numeric_std
-path \
    /home/krystal/vhdl/packages/numeric_std.vhdl
legacy_genus:/hdl_libraries/ieee/packages> ls
./      numeric_std
```

- This example also creates a package named `numeric_std` for the same library. However, the command is issued from the root directory and therefore assumes a run-time penalty:

```
legacy_genus:/> hdl_create package numeric_std -path /home/krystal/vhdl/
packages \
    /hdl_libraries/ieee/packages/numeric_std
legacy_genus:/> ls /hdl_libraries/ieee/packages/
./      numeric_std
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related commands:

hdl_create binding on page 299
hdl_create component on page 301
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create operator on page 306
hdl_create parameter on page 309
hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create parameter

```
hdl_create parameter parameter_name
     [-hdl_invisible]
     [hdl_comp | parameters]
```

Creates a parameter for a ChipWare component. The created parameter will be a `hdl_param` object type and located under `../component_name/parameters`. The default `hdl_parameter` attribute value for parameters created with this command will be `true`. However, if the `-hdl_invisible` option is specified, the default value becomes `false`.

Options and Arguments

hdl_comp | *parameters*	Specifies the path name of the component that holds this parameter.
`-hdl_invisible`	Specifies that the parameter cannot be accessed from the HDL. The value of the `hdl_parameter` attribute for this parameter becomes `false` with this option.
parameter_name	Specifies the name of the parameter that will be created.

Examples

- Both of the following examples create the `WIDTH` parameter for the `CW_sweet_div` component. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command:

```
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div> hdl_create parameter
WIDTH
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div/parameter> ls
./      WIDTH
```

- This example also creates a parameter named `WIDTH` for the same component. However, the command is issued from the root directory and therefore assumes a run-time penalty:

```
legacy_genus:/> hdl_create parameter WIDTH /hdl_libraries/CW/components/
CW_sweet_div
legacy_genus:/> ls /hdl_libraries/CW/components/CW_sweet_div/parameters/
./      WIDTH
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in the *Genus ChipWare Developer Guide for Legacy UI*

- CWD Component
- ChipWare Installation and Registration

Affects this attribute:	hdl_parameter
Related commands:	hdl_create binding on page 299
	hdl_create component on page 301
	hdl_create implementation on page 303
	hdl_create library on page 305
	hdl_create operator on page 306
	hdl_create package on page 307
	hdl_create pin on page 311

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_create pin

```
hdl_create pin pin_name
     {-input | -output | -inout}
     [pins | hdl_oper | hdl_comp]
```

Creates a pin for either a ChipWare component or a synthetic operator. You must specify a pin direction.

Options and Arguments

`-inout`	Specifies that the created pin will be an bidirectional pin.
`-input`	Specifies that the created pin will be an input pin.
`-output`	Specifies that the created pin will be an output pin.
`pin_name`	Specifies the name of the pin that will be created.
*`pins	hdl_oper	hdl_comp`*	Specifies the path name of the component or synthetic operator that holds the created pin.

Examples

- Both of the following examples create the `div_in` input pin for the `CW_sweet_div` component. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command:

```
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div> hdl_create pin -input
div_in
legacy_genus:/hdl_libraries/CW/components/CW_sweet_div/pins/> ls
./      div_in
```

- This example also creates an input pin named `div_in` for the same component. However, the command is issued from the root directory and therefore assumes a run-time penalty:

```
legacy_genus:/> hdl_create pin -input div_in
legacy_genus:/> ls /hdl_libraries/CW/components/CW_sweet_div/pins/
./      div_in
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in the *Genus ChipWare Developer Guide for Legacy UI*

- CWD Component
- ChipWare Installation and Registration
- Synthetic Operator

Related commands:

hdl_create binding on page 299
hdl_create component on page 301
hdl_create implementation on page 303
hdl_create library on page 305
hdl_create operator on page 306
hdl_create package on page 307
hdl_create parameter on page 309

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

7

Input and Output

- decrypt on page 317
- encrypt on page 318
- export_critical_endpoints on page 321
- export_oa_db on page 323
- hdl_set_vlog_file_extension on page 325
- import_oa_db on page 327
- read_clip on page 328
- read_db on page 329
- read_def on page 331
- read_dfm on page 332
- read_dft_abstract_model on page 334
- read_encounter on page 335
- read_hdl on page 336
- read_ilm on page 340
- read_ilm_from_files on page 341
- read_io_speclist on page 342
- read_memory_view on page 343
- read_metric on page 344
- read_netlist on page 345
- read_physical on page 347
- read_pmbist_interface_files on page 348
- read_power_intent on page 349

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- read_qrc on page 350
- read_saif on page 351
- read_sdc on page 352
- read_sdf on page 355
- read_sdp_file on page 356
- read_spef on page 357
- read_tcf on page 358
- read_vcd on page 359
- split_db on page 360
- write_atpg on page 361
- write_bsdl on page 362
- write_compression_macro on page 363
- write_db on page 364
- write_def on page 366
- write_design on page 367
- write_dft_abstract_model on page 369
- write_dft_compression_test_points on page 370
- write_dft_deterministic_test_points on page 371
- write_dft_lbist_test_points on page 372
- write_do_ccd on page 373
- write_do_ccd cdc on page 374
- write_do_ccd compare_sdc on page 375
- write_do_ccd generate on page 377
- write_do_ccd validate on page 381
- write_do_clp on page 382
- write_do_lec on page 385
- write_do_verify on page 390
- write_do_verify cdc on page 391

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- write_encounter on page 393
- write_et_atpg on page 396
- write_et_bsv on page 397
- write_et_lbist on page 398
- write_et_no_tp_file on page 399
- write_ett on page 400
- write_flow on page 401
- write_flow_template on page 402
- write_forward_saif on page 403
- write_hdl on page 404
- write_icgc_monitor on page 408
- write_io_speclist on page 409
- write_ldb on page 410
- write_logic_bist_macro on page 411
- write_metric on page 412
- write_name_mapping on page 413
- write_pmbist_interface_files on page 414
- write_pmbist_testbench on page 415
- write_power_intent on page 416
- write_saif on page 417
- write_scandef on page 418
- write_script on page 419
- write_sdc on page 422
- write_sdf on page 426
- write_sdp_file on page 433
- write_set_load on page 434
- write_snapshot on page 435
- write_spef on page 437

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- write_sv_wrapper on page 438
- write_tcf on page 442
- write_template on page 443
- write_tempus on page 446

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

decrypt

`decrypt [-keydb path] file`

Decrypts and evaluates a Tcl file that was encrypted with the `encrypt` command.

Options and Arguments

file	Specifies the name of the Tcl file to decrypted and evaluated.
-keydb *path*	Sets the `NCPROTECT_KEYDB` environment variable to the directory containing the public key needed to decrypt the file. If you omit this option, you need to set the `NCPROTECT_KEYDB` environment variable before you run the `decrypt` command.

Example

The first command encrypts the Tcl file `my_script.g`. The second command decrypts the `my_script_encr.g` file. These commands are normally executed in different RC sessions.

```
legacy_genus:/> encrypt -tcl my_script.g > my_script_encr.g
legacy_genus:/> decrypt my_script_encr.g
```

Related Information

Related command: encrypt on page 318

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

encrypt

```
encrypt inputfile_name
     [-vlog | -vhdl | -tcl] [-pragma]
     [> file]
```

Uses the NC Protect protection scheme to encrypt the specified HDL or Tcl files.

Tip

To load encrypted HDL files, use the `read_hdl` command.

The command to source an encrypted Tcl file depends on the file extension of the encrypted Tcl file:

- If the encrypted file does not have the `.etf` extension, use the `decrypt` command.
- If the encrypted file has the `.etf` extension, you can use the `source` command.

Options and Arguments

input_file_name	Specifies the file to be encrypted.
file	Specifies the name of the encrypted file. By default, the encrypted file is printed to standard out.
`-pragma`	Specifies to only encrypt the text between the `protect begin` and `protect end` NC Protect pragmas.
`-tcl`	Specifies that the file to be encrypted is a Tcl file. To hide the body of the Tcl scripts, make sure to code procedures using `hidden_proc`.
`-vhdl`	Uses VHDL style comments for NC Protect pragmas.
`-vlog`	Uses Verilog style comments for NC Protect pragmas. This is the default option.

Example

- The following example encrypts the `ksable.vhdl` VHDL file, with VHDL constructs, to a file named `ksable_encrypted.vhdl`. The encrypted file is then loaded.

```
legacy_genus:/> encrypt -vhdl ksable.vhdl > ksable_encrypted.vhdl
legacy_genus:/> read_hdl -vhdl ksable_encrypted.vhdl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example illustrates Verilog code with Verilog style NC Protect pragmas. You must specify `//pragma protect` before specifying the beginning (`//pragma protect begin`) and ending (`//pragma protect end`) pragmas.

```
 module secret_func (y, a, b);
    parameter w = 4;
    input [w-1:0] a, b;
    output [w-1:0] y;
// pragma protect
// pragma protect begin
    assign y = a & b;
// pragma protect end
endmodule
```

  Specifying the `-vlog` and `-pragma` options together will only encrypt the text between the pragmas. The following command encrypts the original Verilog file (`ori.v`) that contained the NC Protect pragmas. The encrypted file is called `enc.v`.

```
legacy_genus:/> encrypt -vlog -pragma org.v > enc.v
```

- The following example illustrates VHDL code with VHDL style NC Protect pragmas. You must specify `--pragma protect` before specifying the beginning (`--pragma protect begin`) and ending (`--pragma protect end`) pragmas.

```
entity secret_func is
    generic (w : integer := 4);
    port (  y: out bit_vector (w-1 downto 0);
        a, b: in bit_vector (w-1 downto 0)      );
end;

-- pragma protect
-- pragma protect begin
architecture rtl of secret_func is
begin
    y <= a and b;
end;
-- pragma protect end
```

  Specifying the `-vhdl` and `-pragma` options together will only encrypt the text between the pragmas. The following command encrypts the original VHDL file (`ori.vhdl`) that contained the NC Protect pragmas. The encrypted file is called `enc.vhdl`:

```
legacy_genus:/> encrypt -vhdl -pragma org.vhdl > enc.vhdl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows a Tcl script (`test.tcl`) with two procedures: the first procedure starting with `proc`, the second one starting with `hidden_proc`. When the script is encrypted, no info will be returned for the `im_hidden` procedure.

```
# pragma protect
# pragma protect begin
proc im_visible {args} {
    # 'info' command will return data for this proc
}
hidden_proc im_hidden {args} {
    # 'info' command will not return data for this proc
}
# pragma protect end
```

```
legacy_genus:/> encrypt -tcl test.tcl > test_enc.tcl
legacy_genus:/> info body im_hidden
legacy_genus:/> info body im_visible

# 'info' command will return data for this proc
```

Related Information

Related command: decrypt on page 317

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

export_critical_endpoints

```
export_critical_endpoints
     -rc_file string -fe_file string
     [-group | -no_group]
     [-percentage_of_endpoints integer]
     [-no_of_bins integer] [-verbose]
     [-percentage_difference integer] [-rtl]
     [-design string] [> file]
```

Generates a path adjust file, which allows synthesis to provide better timing closure results to Innovus.

Options and Arguments

`-design` *`string`*	Specifies the module name.
`-fe_file` *`string`*	Specifies the Innovus slack report that you want to compare.
`file`	Specifies the name of the file to write the report.
`[-group|-nogroup]`	Specifies whether to groups endpoints into bins for path_adjust or not. *Default*: -`group`
`-no_of_bins` *`integer`*	Specifies the number of bins to group the endpoints for compression. *Default:*10 bins each for tighten and relax
`-percentage_difference` *`integer`*	Specifies the percentage difference between the endpoints to be path adjusted (with the path_adjust command). *Default:* 70%
`-percentage_of_endpoints` *`integer`*	Specifies the percentage of endpoints to be constrained or relaxed. *Default:* 20%
`-rc_file` *`string`*	Specifies the Genus endpoint report that you want to compare.
`-rtl`	Writes a path adjust file that can be applied to the RTL.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-verbose`	Specifies a verbose report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

export_oa_db

```
export_oa_db
     -ref_lib string
     [-design_cell string] [-design_view string]
     [-design_lib string] [-lib_path string]
     [-invs_write_design_dir string]
```

Generates the Open Access database which can be loaded back in Innovus™ Implementation System and Virtuoso®.

Options and Arguments

`-design_cell` *`string`*

Specifies the name of the Open Access design cell to export to.

Default: `design_name`

`-design_lib` *`string`* Specifies the name of the Open Access library to export to.

If the library does not exist, the library will be created in the directory specified by the `-lib_path` option.

If the library already exists (as a pointer in cds.lib) and the `-lib_path` option points to another directory, an error will be issued.

Default: `oa_out_`*`process_id`*

`-design_view` *`string`*

Specifies the name of the Open Access view to export to.

Default: `synthesized`

`-invs_write_design_dir` *`string`*

Specifies the directory where the `write_design -innovus` data is saved.

Default: `.oa_export_`*`process_id`*

`-lib_path` *`string`* Specifies the path to the library if the library must be created.

Default: `./oa_export_`*`process_id`*

`-ref_lib` *`string`* Specifies the list of Open Access reference libraries.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: import_oa_db on page 327

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

hdl_set_vlog_file_extension

```
hdl_set_vlog_file_extension
     {-v1995 | -v2001 |-sv} string ...
```

Specifies the mapping between the non-default filename extension(s) and the corresponding Verilog language version.

Following are the default file extensions for the Verilog standards:

- Verilog 1995: `.v95, .v95p`
- Verlog 2001: `.v, .v2k, .vp`
- System Verilog: `.sv, .svp`

Options and Arguments

string	Specifies the filename extension. **Note:** The filename extension must start with the "." character, and is case sensitive.
`-sv`	Specifies that HDL files with the specified extension conform to SystemVerilog P1800.
`-v1995`	Specifies that HDL files with the specified extension conform to Verilog-1995.
`-v2001`	Specifies that HDL files with the specified extension conform to Verilog-2001.

Example

The following `read_hdl` command automatically parses `bot1.v1` in v2001 format, and `test.v2` in System Verilog format.

```
hdl_set_vlog_file_extension -v2001 .v1
hdl_set_vlog_file_extension -sv .v2 .v3
read_hdl -mixvlog bot.v1 test.v2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command: read_hdl on page 336

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

import_oa_db

```
import_oa_db
     -ref_lib string
     -design_lib  string
     -design_cell string
     -design_view string
```

Reads in the Open Access database into Genus and generates the LEF and DEF files.

The generated LEF file is read internally when you execute the `import_oa_db` command.The generated DEF file is read internally post elaboration.

Options and Arguments

`-design_cell` *`string`*	Specifies the Open Access design cell name to be used.import.
`-design_lib` *`string`*	Specifies the Open Access design library name to be used.
`-design_view` *`string`*	Specifies the Open Access design view name to be used.
`-ref_lib` *`string`*	Specifies the list of Open Access reference libraries to import.

Related Information

Related command: export_oa_db on page 323

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_clip

Refer to read_clip in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_db

```
read_db
     {db_file | -from_tcl string} [-mmmc_file file]
     [-quiet] [-verbose]
     [-setup_views] [-hold_views]
     [-dynamic_view] [-leakage_view]
```

Loads the specified database file or Tcl object.

If the database contains setup information, the setup is restored as well. If the setup was written to a separate script, you must source that script before you read the database file.

Note: You should only read a database created with the same tool version. For a database created with an older tool version, use that version of the tool to write out

- the netlist using the `write_hdl` command
- the setup script using the `split_db` command
- all constraint information using the `write_sdc`, `write_cpf`, `write_tcf`, `write_script` commands, and so on

  Next, read these files into the current tool version and regenerate the database if desired.

Options and Arguments

db_file	Specifies the name of the database file to be read.
-dynamic_view	Specifies the `analysis_view` for dynamic power analysis.
-from_tcl *string*	Specifies to read the specified Tcl object.
-hold_views	Specifies a list of hold `analysis_views` for loading.
-leakage_view	Specifies the `analysis_view` for leakage power analysis.
-mmmc_file *file*	Specifies the path name to a Multi-Mode/Multi-Corner (MMMC) configuration file. Use this option if you want to override the settings in the existing MMMC configuration file in the design being restored and instead load a different MMMC file.
-quiet	Suppresses any warning messages. Error messages are printed.
-setup_views	Specifies a list of setup `analysis_views` for loading.
-verbose	Enables verbose output while reading in the database file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related commands: split_db on page 360
write_db on page 364

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_def

Refer to read_def in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_dfm

`read_dfm` *`coefficients_filename`*

Loads the coefficients file. You can only load one file at a time. After the coefficients file is loaded, Genus will annotate the defect probability of any matching cells between the coefficients file and the timing library.

Options and Arguments

`coefficients_filename`

Specifies the name of the coefficients file.

Example

- A DFM file is described in XML format. The following example shows what a DFM file might look like:

```
<?xml version="1.0"?>

<yield_file>
<title> file with probabilities of failure of each library cell </title>

<cell_probability>

  <cell> inv1
    <instance>   0.000000026309750 </instance>
    <systematic> 0.000000000000000 </systematic>
  </cell>

  <cell> fflopd
    <instance>   0.000000153055338 </instance>
    <systematic> 0.000000000000000 </systematic>
  </cell>
  <cell> nand2
    <instance>   0.000000044800000 </instance>
    <systematic> 0.000000000000000 </systematic>
  </cell>

</cell_probability>

</yield_file>
```

- The following example loads two coefficient files:

```
legacy_genus:/> read_dfm test1.dfm
legacy_genus:/> read_dfm test2.dfm
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	report gates -yield
	report yield
Affects this attribute:	yield
Related attribute:	optimize_yield

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_dft_abstract_model

Refer to read_dft_abstract_model in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_encounter

Refer to read_encounter in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_hdl

```
read_hdl file_list
     [ {-v2001 | -v1995 | -sv | -vhdl | -mixvlog}
       [-library library_name[=library_name2]...]
     | -netlist]
     [-f filename] [-define macro=value]... file_list
```

Loads one or more HDL files in the order given into memory. Files containing macro definitions should be loaded before the macros are used. Otherwise, there are no ordering of constraints.

If you do not specify either the `-v1995`, `-v2001`, `-sv` or the `-vhdl` option, the default language format is that specified by the `hdl_language` attribute. The *default* value for the `hdl_language` attribute is `-v2001`.

The HDL files can contain structural code for combining lower level modules, behavioral design specifications, or RTL implementations.

You can automatically read in or write out a compressed HDL file in gzip format. For example:

```
read_hdl sample.v.gz
write_hdl -g sample.v.gz
```

When you load a parameterized Verilog module or VHDL architecture, each parameter in the module or architecture will be identified as an `hdl_param` object and located under `../architecture_name/parameters`. The default `hdl_parameter` attribute value for these parameters will be `true`.

Use the `genus-abort_on_error -f <your script>` command to specify that Genus automatically quit if a script error is detected when reading in HDL files instead of holding at the `genus>` prompt.

Options and Arguments

`-define macro=value`	Defines a Verilog macro with the specified `value`, which is equivalent to the `` `define macro value ``. **Note:** You can also define a macro definition list. **Note:** This option cannot be combined with the `-f` option.
`-f filename`	Specifies the name of the list file for reading files from the simulation environment.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This option is not supported for VHDL designs.

file_list

Specifies the name of the HDL files to load. If several files must be loaded, specify them in a string.

Note: The files can be encrypted.

The host directory where the HDL files are looked for is specified via the `init_hdl_search_path` root attribute.

`-library` *library_name*`[=`*library_name2*`]...`

Specifies the name of the Verilog or VHDL library in which the definitions will be stored.

A virtual directory with this name will be created in the `hdl_libraries` directory of the design hierarchy if it does not already exist.

If you specify multiple libraries, they become multiple names (aliases) of one library. In this case, separate the names with the equal sign (=). Only one of the library names in the list becomes a virtual directory in the `hdl_libraries` directory.

The library definitions remain in effect until elaboration, after which all library definitions are deleted.

By specifying Verilog and VHDL library names, you can read in multiple Verilog modules and VHDL entities (and VHDL packages) with the same name without overwriting each other. See Examples.

Note: You can type `-lib` or `-library`.

`-mixvlog`

Specifies to use the file extension of the Verilog filename(s) to automatically detect the Verilog version of the file.

The mapping between the non-default filename extension and the corresponding Verilog format must have been specified with the `hdl_set_vlog_file_extension` command.

`-netlist`

Reads structural input files when parts of the input design is in the form of a structural netlist. You can read partially structural files provided the structural part of the input design is in the form of structural Verilog-1995 constructs and is contained in separate files from the non-structural (RTL) input.

Note: If this option is specified, all the following options are ignored: `-v1995`,`-v2001`,`-vhdl`,`-sv`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-sv`	Specifies that the HDL files conform to SystemVerilog 3.1.a.
`-vhdl`	Specifies that the HDL files are VHDL files.
`-v1995`	Specifies that the HDL files conform to Verilog-1995.
`-v2001`	Specifies that the HDL files conform to Verilog-2001. This is the default option.
`-vhdl`	Specifies that the HDL files are VHDL files. The `hdl_vhdl_read_version` root attribute value specifies the standard to which the VHDL files conform. *Default*: VHDL-1993

Examples

- The following example first loads the `example1.v` file, then the `example2.v` file:

```
legacy_genus:/> read_hdl {example1.v example2.v}
```

- The following commands with macro definitions are equivalent:
  - `read_hdl -define "A B=4 C"`
  - `read_hdl -define A -define B=4 -define C ...`
- The following command loads a single VHDL file and specifies a single VHDL library.

```
read_hdl -vhdl -library lib1 test1.vhdl
```

- The following commands read structural Verilog files when the design includes RTL (VHDL or Verilog) files:

```
read_hdl file1_bhv.vhdl
read_hdl file2_bhv.v
read_hdl -netlist file3_str.v
elaborate
```

- In the following command, the `-v1995` option is ignored. Both `rtl.v` and `struct.v` are parsed in the structural mode.

```
read_hdl -v1995 rtl.v -netlist struct.v
```

- The following command defines VHDL libraries `lib1`, `lib2` as aliases for `lib3`.

```
read_hdl -vhdl -library lib1=lib2=lib3 test1.vhdl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following commands read in two Verilog files that each contain a Verilog module with the same name (`compute`) but with different functionality. To store both definitions, the `-lib` option indicates in which library to store the definition.

```
read_hdl -v2001 -library lib1 test_01_1.v
read_hdl -v2001 -library lib2 test_01_2.v
```

Inspection of the design hierarchy shows:

```
legacy_genus:/> ls /hdl_libraries/
/hdl_libraries:
./                DP/               DW04/             GB/               STD/              lib2/
AMBIT/            DW01/             DW05/             GTECH/            SYNERGY/          synthetic/
CADENCE/          DW02/             DW06/             IEEE/             SYNOPSYS/
CW/               DW03/             DWARE/            IEEE_SYNERGY/     lib1/

legacy_genus:/> ls /hdl_libraries/lib1/architectures/
/hdl_libraries/lib1/architectures:
./                  compute/
legacy_genus:/> ls /hdl_libraries/lib2/architectures/
/hdl_libraries/lib2/architectures:
./                  compute/
```

Related Information

Reading a Partially Structural Design in *Genus User Guide for Legacy UI.*

Reading Designs in Simulation Environment in *Genus Synthesis Flows Guide for Legacy UI*

Affected by this command:	hdl_set_vlog_file_extension on page 325
Affects this command:	elaborate on page 505
Related command:	read_netlist
Affects this attribute:	hdl_parameter
Affected by these attributes:	init_hdl_search_path
	hdl_language
	hdl_preserve_dangling_output_nets
	hdl_verilog_defines

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_ilm

Refer to read_ilm on page 540 in Chapter 10, “Hierarchical Flow,”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_ilm_from_files

Refer to read_ilm_from_files on page 541 in Chapter 10, “Hierarchical Flow,”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_io_speclist

Refer to read_io_speclist in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_memory_view

Refer to read_memory_view in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_metric

Refer to read_metric in Chapter 4, “Metrics.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_netlist

```
read_netlist file_list
     [-top top_module_name]
     [-define macro=value]...
```

Reads and elaborates a Verilog 2001 structural netlist when the design does not include behavioral (VHDL or Verilog) modules.

A structural Verilog file contains only structural Verilog-2001 constructs, such as module and gate instances, concurrent assignment statements, references to nets, bit-selects, part-selects, concatenations, and the unary ~ operator. Using the `read_hdl -netlist` command uses less memory and runtime to load a structural file than the `read_hdl` command.

- Use the `read_netlist` command to read and elaborate a design and create a generic netlist that is ready to be synthesized. You do *not* need to use the `elaborate` command
- Use the `read hdl -netlist` command to read in a design that *includes* behavioral (VHDL or Verilog) files.

Options and Arguments

`-define` *`macro=value`*

Defines a Verilog macro with the specified *`value`*, which is equivalent to the `` `define macro value ``.

`file_list`

Specifies the name of the HDL files to load. If several files must be loaded, specify them in a string.

The host directory where the HDL files are looked for is specified via the `init_hdl_search_path` root attribute.

`-top` *`top_module_name`*

Specifies the top-level structural Verilog module to be read and elaborated.

If you do not specify this option and multiple top-level modules are found in the loaded netlist, the tool randomly selects one of them and deletes the remaining top-level modules.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related Commands:	read_hdl `-netlist`
	write_hdl on page 404
Sets these attributes:	(design) `hdl_v2001`
	(instance) `hdl_v2001`
	(port) `hdl_v2001`
	subdesign `hdl_v2001`
	subport) `hdl_v2001`
Affected by these attributes:	hdl_preserve_dangling_output_nets

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_physical

Refer to read_physical in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_pmbist_interface_files

Refer to read_pmbist_interface_files in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_power_intent

Refer to read_power_intent in Chapter 15, “Advanced Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_qrc

`read_qrc` *`string...`*

Reads the parasitic extraction technology file.This command should be issued after `read_libs` but before `read_mmmc`. This is used only in the simple mode (without MMMC).

Options and Arguments

`string`	Specifies the name of the parasitic extraction technology file-name (QRC tech file) to be read.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_saif

Refer to read_saif in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_sdc

```
read_sdc file
     [-stop_on_errors] [-no_compress]
     [-echo] [-verbose]
     [-mode mode_name]
```

Reads a constraints file in Synopsys Design Constraint (SDC) format into Genus. Genus creates a cost group for each clock defined in the file. It does not create false paths between these clocks.

You must first elaborate the design before you can read the design constraints.

If you use the `read_sdc` command for loading a subset of timing constraints that includes native Genus commands, such as adding exceptions (`path_delays`), the `write_sdc` command will write these exceptions out in the SDC file.

After using the `read_sdc` command, if you make hierarchy changes in Genus using the `ungroup` or `group` commands, and there are pin specific constraints, then the `write_sdc` command will reflect the change in the hierarchy. For example, if you have constraints on the hierarchy pins you have ungrouped, then the constraints are moved to buffers (that are automatically inserted by Genus when ungrouping). The SDC file will have constraints reflecting these buffers.

Unsupported Constraints

Not all SDCs are supported. For those that are not supported, Genus will issue a warning message but store them for output for the `write_sdc` command. Genus will only store the SDCs and not manipulate any data with them.

The following SDCs are not supported:

```
set_max_area
set_propagated_clock
set_scan_style
set_signal_type
set_test_methodology
set_wire_load_min_block_size
get_references
get_reference
set_connection_class
set_critical_range
set_fix_multiple_port_nets
set_local_link_library
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-echo`	Echoes he SDC commands in the logfile and on the console.
`file`	Specifies the name constraints file to read. You can also specify a file that was compressed with gzip (`.gz` extension).
`-mode` *`mode_name`*	Reads mode-specific constraints for a design.
`-no_compress`	Turns off advanced compression and compression of exceptions for the remainder of the session. *Important* The tool issues message SDC-230 if a potential runtime increase can be expected when you set this option. This message is given when the ratio of the number of exceptions to the number of instances exceeds 0.068, and the number of instances exceeds 100,000.
`-stop_on_errors`	Stops reading the remainder of the script if an error is encountered during reading of the SDC file.
`-verbose`	Displays additional information for each SDC command.

Related Information

Affects this command:	syn_generic on page 526
	create_mode on page 453
Related command:	write_sdc on page 422
Affected by these attributes:	break_timing_paths_by_mode
	enable_break_timing_paths_by_mode
	detailed_sdc_messages
Related attributes:	embedded_script
	enable_data_check
	scale_factor_group_path_weights

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

tim_ignore_data_check_for_non_endpoint_pins

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_sdf

```
read_sdf file
     [-design design]
     {-view view_name}
```

Reads the annotated delays from a file in Standard Delay Format (SDF) and applies it to the specified design for the specified analysis view.

Options and Arguments

`-design design`	Specifies the name of the design to which the annotated delays must be applied.
`file`	Specifies the name of the SDF file to read.
`-view view_name`	Reads the view-specific delays for the design.

Related Information

Related command: write_sdf on page 426

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_sdp_file

Refer to read_sdp_file in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_spef

Refer to read_spef in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_tcf

Refer to read_tcf in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_vcd

Refer to read_vcd in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

split_db

```
split_db
     {input_db_file | -from_tcl file}
     -script file [-to_file file]
```

Reads a database with setup information and writes out the setup information in a setup script and the netlist information into a setup-free database.

If `write_db` command was used without the `-script` option, you can use the `split_db` command to extract the root attributes from the existing database. This command reads the database along with the setup information and then writes out all the root attributes to a separate tcl file specified in the `-script` option. A new database file is also created (specified in the `-to_file` option).

Options and Arguments

`-from_tcl` *`file`*	Specifies to read the specified Tcl object.
`-script` *`file`*	Specifies the name of the script to be written. If this option is specified without the `-to_file` option, the tool writes both the setup and the database to the script. This ensures that the Tcl script and its database are always in sync. **Note:** Storing the database in the script takes more memory and results in a larger file.
`-to_file` *`file`*	Specifies the name of the new database file.
`input_db_file`	Specifies the name of the database file to be read.

Examples

```
split_db my_org.db -to_file my_new.db -script my_new.root_attribute.tcl
```

Related Information

Related commands: read_db on page 329
write_db on page 364

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_atpg

Refer to write_atpg in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_bsdl

Refer to write_bsdl in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_compression_macro

Refer to write_compression_macro in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_db

```
write_db
     [-all_root_attributes] [-no_root_attributes]
     [db_file] [-script file] [-design design]
     [-quiet] [-verbose]
```

This command writes out a netlist, SDC, physical information (if any), power intent information (if any) and root attributes to a database file or returns a Tcl list. But, it doesn't write out any user defined variables. By default, only root attributes affecting the QOR are written out. Root attributes with non-default settings can be included in the database or written to a script if the `-all_root_attributes` option is specified.

To read this database in a different environment, you need to specify the `-script` option which helps to write all root attributes in a separate tcl file, which can be later, edited to change the library paths.

Options and Arguments

`-all_root_attributes`
: Writes out all the root attributes with non-default settings.

`db_file`
: Specifies the name of the database file to be written.

  By default, the command returns a Tcl list.

`-design` *`design`*
: Specifies the design for which to write out the database information.

  If omitted, the design defaults to the current design.

`-no_root_attributes`
: Prevents writing out of the root attributes with non-default settings.

  You cannot specify this option with the `-script` option.

`-quiet`
: Suppresses any warning messages. Error messages are printed.

`-script` *`file`*
: Includes all root attribute settings in the specified script.

  If this option is omitted, this information is included in the database file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If this option is specified without the `-to_file` option, the tool writes both the setup and the database to the script. This ensures that the Tcl script and its database are always in sync.

Note: Storing the database in the script takes more memory and results in a larger file.

`-verbose` Enables verbose output while writing out the database file.

Example

In the following script, the variable `db` is defined for the Tcl object written out by the `write_db` command:

```
set db [write_db /designs/test]
...
read_db -from_tcl $db
```

Related Information

Related commands:

define_attribute on page 1371
read_db on page 329
split_db on page 360
write_design on page 367

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_def

See write_def on page 837 in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_design

```
write_design
     [-base_name string] [-gzip_files]
     [-innovus] [-hierarchical] [-db] [design]
```

Generates all the files needed to reload the session in Genus (for example, `.g`, `.v`. and `.tcl` files). If you want to generate all the files that are need to loaded in both a Genus and Innovus® session, use the `-innovus` option to write innovus specific files. To reload the session in Genus, `source *.genus_setup.tcl` in Genus shell . To reload session in Innovus, `source *.invs_setup.tcl` in the Innovus shell.

Note: When you use this command on a design that is not fully mapped, for example after `elaborate` or `syn_generic`, and then reload and map the design, the final area and timing results may differ from the results obtained in a single synthesis session.

Options and Arguments

`-basename` *`string`*	Specifies the path and basename for the generated files. The setup files will be written using the relative/absolute paths as provided through this.
`-db`	Specifies to write out Innovus DB file by invoking Innovus. **Note:** This option can only be used when `-innovus` is also used
`design`	Specifies the top-level design in Genus.
`-gzip_files`	Compresses the generated files in gzip format. **Note:** Since Global Timing Debug (in Encounter Timing System) does not handle compressed SDC files, the `write_design` command will not compress SDC files.
`-hierarchical`	Writes out additional information for the hierarchical flow using interface logic models.
`-innovus`	Generates the additional files needed for Innovus.

Example

- The following example writes both the Genus and Innovus files as well as specifies the path and basename to be `test/top`. To reload the session in Genus, source

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`top.genus_setup.tcl` in Genus shell. To reload the session in Innovus, source `top.invs_setpu.tcl` in the Innovus shell.

If absolute path is not provided, invoke the session from the same directory where previous session was invoked from.

```
legacy_genus:/> write_design -basename test/top
```

```
unix> ls test
top.matrics.json
top.v
top.spl.etf
top.cmap.gz
top.def
top.g
top.genus_setup.tcl
```

```
legacy_genus:/> write_design -basename /foo/foo1/test/top  ; # Will write out
the setup files with absolute paths as "-basename" is provided with absolute
path
```

```
unix> ls test
top.matrics.json
top.v.gz
top.spl.etf.gz
top.cmap.gz
top.def.gz
top.mode
top.size_ok.tcl
top.flowkit_settings.tcl
top._default_constraint_mode_.sdc
top.wnm_attrs.tcl
top.mmode.tcl
top.ndr.tcl
top.min_layer
top.globals
top.invs_setup.tcl
top.genus_setup.tcl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_abstract_model

Refer to write_dft_abstract_model in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_compression_test_points

Refer to write_dft_compression_test_points in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_deterministic_test_points

Refer to write_dft_deterministic_test_points in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_lbist_test_points

Refer to write_dft_lbist_test_points in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd

```
write_do_ccd {cdc | compare_sdc | generate | propagate | validate}
```

Translates Genus settings to Conformal's Constraint Designer (CCD) commands for the *Validate* and *Generate* flows. In the *Validate* flow, by default the command compares the SDC to the RTL

Options and Arguments

`cdc`	Generates a dofile for Clock Domain Crossing Checks.
`compare_sdc`	Generates a dofile to compare two SDC files.
`generate`	Generates a dofile for the *Generate* flow.
`propagate`	Generates a dofile to create a chip-level SDC file.
`validate`	Generates a dofile for the *Validate* flow.

Related Information

Related commands:

write_do_ccd cdc on page 374
write_do_ccd compare_sdc on page 375
write_do_ccd generate on page 377
write_do_ccd propagate on page 379
write_do_ccd validate on page 381

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd cdc

```
write_do_ccd cdc
     [-design string] [-sdc files]
     [-no_exit] [-logfile file] [> file]
```

Writes a dofile for the Conformal® Constraint Designer (CCD) for clock domain crossing checks.

Options and Arguments

`-design` *`string`*	Specifies the top-level design in Genus. If omitted, the design defaults to the current design. This option is required if multiple designs are loaded in the session.
`file`	Specifies the file to which the report must be written.
`-logfile` *`file`*	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
`-no_exit`	Suppresses the `EXIT` command at the end of the dofile.
`-sdc` *`files`*	Specifies the SDC files to be read. You can omit this option if you read in the SDC files with the `read_sdc` command. An error is given when no SDC files are read in either through this command or through `read_sdc`.

Example

```
write_do_ccd cdc -sdc test.sdc -design test
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd compare_sdc

```
write_do_ccd compare_sdc
     [-design string] [-netlist file]
     -golden_sdc file -revised_sdc file
     [-pre_load script] [-pre_exit script]
     [-no_exit] [-logfile file] [-detail] [> file]
```

Writes a dofile for the Conformal® Constraint Designer (CCD) to compare two SDC files and report any differences between the two files.

Options and Arguments

`-design` *`string`*	Specifies the top-level design in Genus.
`-detail`	Requests a detailed comparison report.
`file`	Specifies the file to which the report must be written.
`-golden_sdc` *`file`*	Specifies the UNIX path to the golden SDC file.
`-logfile` *`file`*	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
`-netlist` *`file`*	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
`-no_exit`	Suppresses the `EXIT` command at the end of the dofile.
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the CCD `EXIT` command.
`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-revised_sdc` *`file`*	Specifies the UNIX path to the revised SDC file.

Example

The following command compares the `test.sdc` and `revised.sdc` files.

```
write_do_ccd compare_sdc -golden_sdc test.sdc -revised_sdc revised.sdc
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: compare_sdc on page 568

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd generate

```
write_do_ccd generate
     [-design string] [-netlist string]
     -in_sdc files [-out_sdc file]
     [-slack integer] [-report file]
     [-fpgen | -dfpgen | -trv]
     [-pre_load script] [-pre_exit script]
     [-no_exit] [-logfile file] [-mode string] [> file]
```

Writes a dofile for the Conformal ® Constraint Designer (CCD) for the *Generate* flow, which generates additional false paths based on critical path timing reports.

Options and Arguments

`-design` *`string`*	Specifies the top-level design in Genus.
`file`	Redirects all the output to the specified file.
`-dfpgen`	Generates a dofile for the directed false path flow.
`-fpgen`	Generates a dofile for the false path flow.
`-in_sdc` *`files`*	Specifies the list of SDC files to load.
`-logfile` *`file`*	Specifies the name of the CCD logfile.
`-mode` *`string`*	Specifies to write out mode-specific constraints for the design. If you omit this option and you did not specify any input SDC files, the command will write out the constraints for all the modes.
`-netlist` *`string`*	Specifies the UNIX path to the netlist. This option compares the SDC to the specified netlist instead of the RTL.
`-no_exit`	Suppresses the `exit` command at the end of the dofile.
`-out_sdc` *`file`*	Specifies the filename to which the identified false paths will be written. *Default*: `cfp.sdc`
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the CCD `exit` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-report` *`file`*	Specifies the name of the timing report file to be generated. The report will be in CCD format.
`-slack` *`integer`*	Specifies the slack value in picoseconds. Only paths below this slack value will be used to generate the timing report. This option should be used with the `-report` option.
`-trv`	Generates a dofile for the timing report validation flow.

Related Information

Affected by this attribute: wccd_threshold_percentage

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd propagate

```
write_do_ccd propagate
     [-design design] [-netlist string]
     -block_sdc string [-glue_sdc string]
     [-partial_chip_sdc string]
     [-out_sdc string]
     [-rule_instance_file string]
     [-rule_instance_template string]
     [-pre_load script] [-pre_exit script]
     [-no_exit] [-logfile string] [> file]
```

Generates a dofile for the Conformal® Constraint Designer (CCD) to propagate block-level constraints to the top-level and integrate them with the glue constraints to generate a chip-level SDC file.

Options and Arguments

`-block_sdc` *`string`*	Specifies a list of block names with their associated block-level SDC files in the following format: `{{block_name block_sdc_file}...}`
`-design` *`string`*	Specifies the top-level design in Genus.
`file`	Specifies the file to which the report must be written.
`-glue_sdc` *`string`*	Specifies the name of a glue SDC file. This file contains a set of constraints for the top-level module only (without covering any block-level constraints).
`-logfile` *`string`*	Creates a separate CCD logfile. You must specify the UNIX path to the file.
`-netlist` *`string`*	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
`-no_exit`	Suppresses the `exit` command at the end of the dofile.
`-out_sdc` *`string`*	Specifies the name of the constraints file that is generated after propagation and integration of the block and glue constraints. *Default*: `chip.sdc`
`-partial_chip_sdc` *`string`*	Specifies the name of the partial SDC file that corresponds to the top-level of the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-pre_exit` *`string`*
: Specifies the name of the dofile (script) that must be sourced before the CCD `exit` command.

`-pre_read` *`string`*
: Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.

`-rule_instance_file` *`string`*
: Specifies the name of the file which defines the rule instances for the Conformal ® Constraint Designer (CCD) tool.

`-rule_instance_template` *`string`*
: Creates a template integration rule instances file.

  You can modify this file according to the design. To use this file as input for the Conformal ® Constraint Designer (CCD) tool, specify the file as value of the `-rule_instance_file` option.

Example

- The following command creates a do file to propagate the block-level SDC files `i1.sdc` and `i2.sdc` to the top level.

```
legacy_genus:/> write_do_ccd propagate -block_sdc {{i1 i1.sdc} {i2 i2.sdc}} \
-out_ sdc ./my_chip.sdc  rule_instance_file my_rules -logfile ccd.log
```

  The generated dofile will be similar to:

```
read library -statetable -liberty ./slow.lib
add search path -design .
read design -verilog ./ti1.v -lastmod -noelab
elaborate design
dofile ./my_rules
read hierarchical sdc \
-sdc_design i1 i1.sdc \
-sdc_design i2 i2.sdc
set system mode verify
integrate -all ./chip.sdc -replace
report rule check
report environment
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_ccd validate

```
write_do_ccd validate
     [-design string] [-netlist string]
     -sdc string
     [-init_sequence_file string]
     [-pre_load script] [-pre_exit script]
     [-no_exit] [-logfile string] [> file]
```

Writes a Conformal Constraint Designer (CCD) dofile for the *Validate* flow, which validates the constraints and false path exceptions.

Options and Arguments

`-design` *`string`*	Specifies the top-level design in Genus.
`file`	Redirects all the output to the specified file.
`-init_sequence_file` *`string`*	Specifies the UNIX path to the initialization sequence file for multi-cycle path validation.
`-logfile` *`string`*	Specifies the name of the CCD logfile.
`-netlist` *`string`*	Specifies the UNIX path to the netlist. This option compares the SDC to the specified netlist instead of the RTL.
`-no_exit`	Suppresses the `exit` command at the end of the dofile.
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the CCD `exit` command.
`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-sdc` *`string`*	Specifies the list of SDC files.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_clp

```
write_do_clp
     [-design design] [-netlist string ]
     [-env_var string]
     [-add_iso_cell string] [-clp_out_report string]
     [-ignore_ls_htol] [-verbose]
     [-pre_read script] [-pre_exit script]
     [-cpf_file file | -files_1801 file]
     [-all | -lp_only]
     [-no_exit] [-tmp_dir string] [-logfile file]
     [-tclmode] [> file]
```

Writes the required dofile for Conformal Low Power (CLP).

Note: This command will issue an error and will not proceed if you have multiple Common Power Format (CPF) files.

For more information on the Low Power Rule (CLP) Checks, refer to the Conformal® Low Power Reference Manual.

Options and Arguments

`-add_iso_cell` *string*
: Specifies the standard cells that CLP should recognize as isolation cells.

`-all`
: Specifies to perform all rule checks, whether low-power related or not. This is also the default behavior.

`-clp_out_report` *string*
: Writes the output of the `report rule check` Conformal® Equivalence Checking command to this specified file.

`-cpf_file` *file*
: Specifies the CPF file to be used for low power checks.

`-design` *design*
: Specifies the top-level design in Genus.

`-env_var` *string*
: Specifies the names and values of UNIX environment variables to be used in the library, design, and logfile names in the generated dofile.

file
: Redirects all the output to the specified file.

`-files_1801` *file*
: Specifies the 1801 file to be used for low power checks.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-ignore_ls_htol`	Indicates whether to ignore the high to low level shifter check. If this option is specified, the following CLP directive will be added to the dofile: `set lowpower option -ignore_high_to_low`
`-logfile` *`file`*	Specifies the name of the CLP logfile.
`-lp_only`	Specifies to only perform the low power rule check errors and warnings. By default, non low-power related issues, such as structural issues are reported as well.
`-netlist` *`path`*	Specifies the UNIX path that contains the netlist containing all the design's low power features (for example, level shifters, isolation cells and SRPG flops). *Default*: RTL
`-no_exit`	Indicates whether to skip the `exit` command at the end of the dofile. If this option is specified, the following command will be omitted from the dofile: `exit -force`
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the CLP `exit` command.
`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-tclmode`	Specifies to generate the dofile as a Tcl script.
`-tmp_dir` *`string`*	Specifies the name of the directory to which the generated files must be written. Its contents will be CLP native commands. *Default*: `RC_CLP_`*`design_name`*`_out.do`
`-verbose`	Indicates whether the generated dofile will be verbose. If this option is specified, the `report rule check` command should write out the intermediate dofile for CLP and then include that file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following is an example of a CLP dofile:

```
set log file log_file_name -replace
set lowpower option -netlist_style logical

read library -statetable -liberty bn65lplvt_121a/tcbn65lplvtwcl0d90d72.lib \
read design -verilog -sensitive netlist.v

read cpf file cpf_file_name

analyze power domain
rep rule check ISO* LSH* RET* -verbose
exit -force
```

Related Information

Affected by these attributes: wclp_lib_statetable

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_lec

```
write_do_lec [-top string]
     [-golden_design {rtl|fv_map|file}]
     -revised_design {fv_map|file}
     [-1801_golden file] [-1801_revised file]
     [-cpf_golden file] [-cpf_revised file]
     [-no_lp] [-no_insert_iso_in_dof]
     [-sim_lib string] [-sim_plus_liberty] [-cw_sim string]
     [-no_dft] [-dft_constraint_file file]
     [-log_file file] [-env_var string]
     [-pre_read file] [-pre_compare file]
     [-pre_exit file] [-no_exit]
     [-hierarchical_reference | -flat]
     [-write_session file] [-checkpoint file]
     [-tmp_dir string] [-verbose] [> file]
```

Writes a script and verification information that can by used Conformal® Logical Equivalence Checking tool to compare the specified golden and revised designs.

This command works with the Common Power Format (CPF) flow: if it detects a CPF file then it will output this information to the Conformal LEC dofile.

Options and Arguments

`-1801_golden` *file*	Specifies the name of the golden (original) 1801 file. **Note:** This option only applies to the IEEE-1801 flow.
`-1801_revised` *file*	Specifies the name of the revised 1801 file that corresponds to the current state of the design. **Note:** This option only applies to the IEEE-1801 flow.
`-checkpoint` *file*	Specifies the name of the checkpoint file to generate.
`-cpf_golden` *file*	Specifies the name of the golden (original) file. If omitted, the file will be inferred from the `cpf_files` design attribute. **Note:** This option only applies to the CPF flow.
`-cpf_revised` *file*	Specifies the name of the revised CPF file. If you omit this option, the file specified for the `-cpf_golden` option will be used.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you expect name changes in the CPF design objects, you must generate this file using the `write_power_intent -cpf` command.

Note: This option only applies to the CPF flow.

`-cw_sim` *`string`*

Specifies the language of the simulation models that the Conformal Logical Equivalence Checker should use to infer the ChipWare components.

You can specify any of the following values:

`{V1995|V2001|SV|VHDL1987|VHDL1993}`

Default: Language specified in the `read_hdl` command.

`-dft_constraint_file` *`file`*

Sources DFT constraints from the specified file before executing the `compare` command.

`-env_var` *`string`*

Specifies the names and values of UNIX environment variables to be used for directory and file names in the dofile.

`file`

Redirects all the output to the specified file.

`-flat`

Performs a flattened Conformal LEC comparison.

`-golden_design {rtl|fv_map|`*`file`*`}`

Specifies the golden design to compare to. You can specify the following values:

`rtl`—`write_do_lec` will access the `hdl_filelist` attribute to use the same set of files that were read into Genus.

`fv_map`—`write_do_lec` will use the netlist automatically written by Genus during mapping (`syn_map`).

`file`—Specifies the UNIX path to an alternative golden design.

If the file was loaded into Genus (using either `read_hdl` or `read_netlist`), the tool knows the language format of the file. Otherwise, the tool assumes that the format of the file is Verilog-1995.

Default: `rtl`

`-hierarchical_reference`

Performs a hierarchical Conformal LEC comparison.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-log_file` *`file`*	Specifies the name of the Conformal LEC logfile.
`-no_dft`	Prevents that DFT constraints are written to the dofile.
`-no_exit`	Does not add the `exit` command to the end of the dofile.
`-no_insert_iso_in_dof`	Prevents that the `-insert_isolation` option is added to the LEC `commit power intent` commands for the golden and revised design in the generated dofile. Use this option when you write out a dofile before the power intent is committed in Genus (before using the `commit_power_intent` command)
`-no_lp`	Prevents verification of power intent information.
`-pre_compare` *`file`*	Specifies an extra dofile that must be sourced from the dofile before the LEC `set system mode lec` or `write hier_compare dofile` command.
`-pre_exit` *`file`*	Specifies an extra dofile that must be sourced from the dofile before the LEC `exit` command.
`-pre_read` *`file`*	Specifies an extra dofile that must be sourced from the dofile before the library and design are read.
`-revised_design {fv_map|`*`file`*`}`	Specifies the revised design to compare to the golden design. You can specify the following values: `fv_map`—`write_do_lec` will use the netlist automatically written by Genus during mapping (`syn_map`). *`file`*—Specifies the UNIX path to the revised design.
`-sim_lib` *`string`*	Specifies the simulation library in Verilog 1995.
`-sim_plus_liberty`	Specifies that the simulation library is an addition to the synthesis library.
`-smart`	Enables advanced SMART Lec capabilities. **Note:** Requires a Conformal SMARTLec license.
`-tmp_dir` *`string`*	Specifies the name of the directory to which the generated files must be written.
`-top` *`string`*	Specifies the name of the top-level design in Genus.
`-verbose`	Generates a dofile with verbose reporting.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-write_session file` Specifies the filename to save the LEC session.

Examples

- The following command will source the `extra_settigs.do` dofile before the `compare` command in the sample.do file.

```
write_do_lec -revised revised.v -pre_compare extra_settings.do > sample.do
```

- The following command generates a dofile that will point to the System Verilog simulation models of all ChipWare components used in the design.

```
write_do_lec -revised_design ../netlist/top_nl.v -cw_sim sv \
    -log_file cw_sim_sv.log > cw_sim.sv_lec.do
```

Related Information

Interfacing with Conformal Logical Equivalence Checker in *Genus Interface to Conformal for Legacy UI*

Performing a Low Power Equivalence Check in *Genus Interface to Conformal for Legacy UI*

More About the write_do_lec dofile in *Genus Interface to Conformal for Legacy UI*

Related command:	write_power_intent on page 1292
Affected by these attributes:	boundary_optimize_invert_hpins
	dp_ungroup_during_syn_map
	wlec_add_noblack_box_retime_subdesign
	wlec_analyze_abort
	wlec_analyze_setup
	wlec_auto_analyze
	wlec_compare_threads
	wlec_composite_compare
	wlec_cut_point
	wlec_hier_comp_threshold
	wlec_lib_statetable
	wlec_low_power_analysis

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

wlec_set_cdn_synth_root

wlec_uniquify

wlec_use_lec_model

wlec_use_smart_lec

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_verify

```
write_do_verify {cdc}
```

Generates a dofile for Conformal® Verify.

Options and Arguments

`cdc`	Generates a dofile for Clock Domain Crossing Checks

Related Information

Related commands	write_do_verify cdc on page 391

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_do_verify cdc

```
write_do_verify cdc {-categorize | -validate}
     -sdc string
     [-design string] [-no_exit]
     [-logfile string] [> file]
```

Generates a dofile for Conformal® Extended Checks to perform clock domain crossing checks on clock domain crossings in either the Categorization or Validation flow.

- In the Categorization flow, no synchronization rules are defined. Genus automatically identifies and categorizes the clock domain crossing paths.
- In the Validation flow, you define the synchronization rules that specify the valid synchronization structures in the design.

Options and Arguments

`-categorize`	Specifies to generate a dofile for the *Categorization* flow.
`-design` *`string`*	Specifies the name of the top-level design in Genus.
`file`	Specifies a specific dofile filename.
`-logfile` *`string`*	Specifies a specific logfile name.
`-no_exit`	Suppresses the `exit` command in the dofile.
`-sdc` *`string`*	Specifies a list of the SDC files.
`-validate`	Specifies to generate a dofile for the *Validation* flow.

Examples

- The following example writes a dofile for the Categorization flow:

```
write_do_verify cdc -sdc /home/test/general.sdc -logfile my.log \
    -categorize
```

- The following example writes a dofile for the Validate flow:

```
write_do_verify cdc -sdc /home/test/general.sdc -logfile my.log \
    -validate
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these attributes: wcdc_clock_dom_comb_propagation
wcdc_synchronizer_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_encounter

```
write_encounter design [-basename string] [-tcf]
     [-reference_globals_file globals_file |
     -reference_config_file config_file]
     [-floorplan string] [-lef lef_files]
     [-ignore_scan_chains] [-ignore_msv] [-tcf]
     [-gzip_files] [design]
```

Writes Innovus input files to a single directory. The command will only convert library domains into power domains for Innovus if power domains exist in Genus. If power domains do not exist, the `-ignore_msv` option is implied. The command also supports the Common Power Format (CPF) files by directly passing them to Innovus.

The generated files are all required Innovus input files and include the following files:

- Netlist (`.v`)
- Innovus global variables file (.globals) or Innovus configuration file (`.conf`)
- SDC constraints (`.sdc`)
- Tcl script (`.enc_setup.tcl`)
- Mode file (`.mode`)
- Scan DEF file (`.scan.def`)
- MSV-related files (`.msv.tcl`, `.msv.vsf`)
- Multiple timing mode (`.mmode.tcl`)
- Updated CPF file (for a CPF-based flow)

The `.enc_setup.tcl` file can simultaneously load all the necessary Innovus data in an Innovus session. This eliminates the need to load each of the necessary files sequentially.

The `.mode` file contains all the Innovus `setMode` settings. For example, the file would contain the `setAnalysisMode` and `setPlaceMode` settings.

The full DEF file that is outputted is the exact same DEF file that was loaded or generated by `syn_opt physical`. However, Genus generates the information for the Scan DEF file (`.scan.def`).

The updated CPF file contains the power intent of the golden CPF but with updated references to the design objects.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-basename` *`string`*
: Specifies the directory path name and base filename for the output data.

`design`
: Specifies a particular design for which to write out information. Only one design can be specified at a time.

`-floorplan` *`string`*
: Specifies the extension for the file containing the floorplan. The valid extensions are:

  `.def`—DEF

  `.pde`—PDEF

  `.fp`—Innovus floorplan

`-gzip_files`
: Compresses the netlist and constraints in `.gz` format. The floorplan, if one was read, will be untouched. That is, if it was read in uncompressed, it will be outputted uncompressed and vice versa.

  Note: Since Global Timing Debug (in Tempus) does not handle compressed SDC files, the `write_encounter` command will not compress SDC files.

`-ignore_scan_chains`
: If specified, the scan DEF file will not be written and the scan reorder directives will not be included in the setup file.

`-ignore_msv`
: If specified, the MSV setup file and the shifter table file will not be written out. This option is useful if the library domains in Genus are not being used for modeling power domains.

`-lef` *`lef_files`*
: Specifies a particular physical library or libraries to use. The physical libraries will have the `.lef` extension. The contents of the LEF library that was specified with the `lef_library` attribute will be used if this option is not specified.

`-reference_config_file` *`config_file`*
: Specifies a reference Innovus configuration file to use as a template for the generated configuration file.

  Note: The configuration file is used in the Legacy Configuration Data Flow.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-reference_globals_file` *`globals_file`*

Specifies a reference Innovus globals file to use as a template for the generated global variables file.

Use this option with the init_design simple data flow in Innovus.

`-tcf`

Specifies to write out a TCF containing the asserted switching activities of the pins in the design.

Examples

- The following example writes all the Innovus input files for the `test07` design to the directory `TEST`. The basename for the files are specified to be `test1`.

```
legacy_genus:/> write_encounter design test07 -basename TEST/test1

unix> ls TEST/
test1.conf   test1.enc_setup.tcl   test1.mode   test1.sdc   test1.v
```

- The following example compresses the netlist and constraints with the `-gzip_files` option:

```
legacy_genus:/> write_encounter design -gzip_files

unix> ls rc_enc_des
rc.conf  rc.def  rc.enc_setup.tcl  rc.mode  rc.sdc.gz  rc.v.gz
```

Related Information

Related commands: create_mode on page 453
read_encounter on page 335

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_atpg

Refer to write_et_atpg in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_bsv

Refer to write_et_bsv in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_lbist

Refer to write_et_lbist in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_no_tp_file

Refer to write_et_no_tp_file in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_ett

```
write_ett
     [-strict | -dc | -ett]
     [-version {1.1|1.3|1.4}]
     [design] [> file]
```

Generates constraints for Encounter® True Time.

Some constraints (such as `set_input_delay`, `set_output_delay`, and so on) are written out in SDC (Synopsys Design Constraints) format while others (such as `set_false_path`, `set_disable_timing`) are written out in Encounter Test format.

Options and Arguments

design	Specifies the design for which the constraints must be generated.
file	Specifies the name of the file to which the constraints must be written.
-dc	Writes the constraints that are DC and PT compatible, which means that commands not listed in the SDC specification may be written out.
-ett	Writes an Encounter True Time Clock Constraints file.
-strict	Writes only constraints in SDC format.
-version {1.1|1.3|1.4}	Specifies the SDC version to use to write SDC constraints.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_flow

Refer to write_flow in Chapter 3, “Flowkit.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_flow_template

Refer to write_flow_template in Chapter 3, “Flowkit.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_forward_saif

Refer to write_forward_saif in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_hdl

```
write_hdl  {design|subdesign}...
     [-suffix string]
     [-abstract] [-generic] [-depth integer]
     [-equation] [-pg] [-language {v1995|v2001}]
     [-lec] [-use_eeq_cell_with_liberty_info][ > file]
```

Generates one of the following design implementations in Verilog format:

- A structural netlist using generic logic
- A structural netlist using mapped logic

You can automatically read in or write out a gzip compressed Verilog file. For example:

```
read_hdl sample.v.gz
write_hdl > sample.v.gz
```

Options and Arguments

`-abstract`	Generates an empty top-level Verilog module definition of the specified design or subdesign that defines the I/O pins and bit-width for all top-level functional and scan-related ports in the design or subdesign. This empty module description is further referred to as *logic abstract model*.
`-depth` *`integer`*	Specifies the number of hierarchy levels to be written out, starting from the top level. A value of 0, writes out only the top-level module. *Default*: infinite
`{`*`design`* `|` *`subdesign`*`}`	Specifies the design or subdesign for which the design implementation must be generated.
`-equation`	Writes out a logic equation in an assign statement for each Verilog primitive gate.
`file`	Specifies the file to which the output must be written. *Default*: Output is written to the screen.
`-generic`	Generates an unoptimized generic logic implementation of the design that uses the generic logic gates specified within the Verilog language.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Any parts of the design that are mapped will be unmapped for the `write_hdl` command without affecting the design in memory.

Once the synthesize commands (`syn_generic`, `syn_map`, or `syn_opt`) have been run, you cannot recover the version of the design that was generated using this option prior to synthesis.

`-language {v1995| v2001}`

Specifies the language of the output Verilog file. In `v2001`, the tool writes out the Verilog-2001 attributes stored with instances, ports and subports in the design.

Default: `v1995`

`-lec`

Generates an *intermediate* netlist with additional information to facilitate formal verification with Conformal® Equivalence Checking.

`-pg`

Adds the power and ground nets to the netlist, where PG pins, net connections have come through 1801, UPF files.

If the PG pins are coming from a Liberty file, and are connected in RTL, they will be written out by default in the Verilog netlist generated by Genus.

`-suffix`

Specifies the string to be appended to the name of all defined modules in the generated netlist.

`-use_eeq_cell_with_liberty_info`

Specifies whether the Verilog Writer should write the eeq master equivalent or not.

Examples

- The following example writes out the logic abstract model definition of design `test`:

```
legacy_genus:/> write_hdl -abstract

// Generated by Genus Synthesis Solution version

module test(in1, in2, out1, out2, clk1, clk2, clk3, se1, se2);
  input [3:0] in1;
  input [7:0] in2;
  input clk1, clk2, clk3, se1, se2;
  output [3:0] out1;
  output [7:0] out2;
endmodule
```

- The following example writes out the design as generic logic regardless of its current mapped state:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> write_hdl -generic > design_rtl.v
```

- You can write out a netlist for a specific module. For example, the following commands writes out the `middle` module:

```
legacy_genus:/> set_attr unresolved true [get_attr instance [get_attr
subdesign bottom]]
legacy_genus:/> write_hdl [find / -subdesign middle]
```

- The following example writes out a design that instantiates cells from the target technology library reflecting the current state of the design (mapped state):

```
legacy_genus:/> read_hdl design.v
legacy_genus:/> ...
legacy_genus:/> syn_map
legacy_genus:/> ...
legacy_genus:/> write_hdl
```

- The following example replaces each Verilog primitive gate by an equivalent Verilog assign statement:

```
legacy_genus:/> write_hdl -equation design.v
```

- The following example writes out the design as generic logic regardless of its current mapped state:

```
legacy_genus:/> write_hdl -generic > design_rtl.v
```

Related Information

Affected by these commands:	elaborate on page 505
	syn_generic on page 526
Affected by these attributes:	optimize_merge_flops
	optimize_merge_latches
	optimize_merge_seq
	write_sv_port_wrapper
	write_vlog_bit_blast_bus_connections
	write_vlog_bit_blast_constants
	write_vlog_bit_blast_mapped_ports
	write_vlog_bit_blast_tech_cell
	write_vlog_convert_onebit_vector_to_scalar
	write_vlog_declare_wires
	write_vlog_empty_module_for_logic_abstract

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_vlog_line_wrap_limit
write_vlog_no_negative_index
write_vlog_port_association_style
write_vlog_preserve_net_name
write_vlog_top_module_first
write_vlog_unconnected_port_style
write_vlog_wor_wand

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_icgc_monitor

Refer to write_icgc_monitor in Chapter 17, “Joules.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_io_speclist

Refer to write_io_speclist in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_ldb

```
write_ldb
     library outputfile -opt_level {0|1}
```

Writes out a compiled library.

Note: You cannot read the compiled library back in the same session.

Options and Arguments

library	Specifies the full path of the library that must be converted in ldb format.
-opt_level{0|1}	Specifies whether to append the original library in gzip format to the compiled library. ■ 0 appends the source library in zipped format to the compiled library ■ 1 does not append the source library *Default*: 0
outputfile	Specifies the file to which the converted library must be written.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_logic_bist_macro

Refer to write_logic_bist_macro in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_metric

Refer to write_metric in Chapter 4, “Metrics.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_name_mapping

```
write_name_mapping
     [-to_file file]
```

Writes out the RTL name and the corresponding netlist name to a file for each of the following netlist objects:

- sequential instances
- input and output ports
- blackboxes (timing models and unresolved references)

By default, the format of the name mapping information is Conformal-compatible:

`add mapped point RTL_instance_path netlist_instance_path phase_inversion`

For phase inverted flops, `-invert` is added.

Options and Arguments

`-to_file file`	Specifies the name of the file to which the name mapping information must be written. *Default*: `./name_map.rpt`

Example

```
add mapped point iCORE/digit_out[0] iCORE/digit_out_reg[0]_iCORE_digit_out_reg[1]_iCORE_digit_out_reg[2]_iCORE_digit_out_reg[3]/Q1N -invert
add mapped point iCORE/digit_out[1] iCORE/digit_out_reg[0]_iCORE_digit_out_reg[1]_iCORE_digit_out_reg[2]_iCORE_digit_out_reg[3]/Q2 -invert
add mapped point iCORE/digit_out[2] iCORE/digit_out_reg[0]_iCORE_digit_out_reg[1]_iCORE_digit_out_reg[2]_iCORE_digit_out_reg[3]/Q3N
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_pmbist_interface_files

Refer to write_pmbist_interface_files in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_pmbist_testbench

Refer to write_pmbist_testbench in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_power_intent

Refer to write_power_intent in Chapter 15, “Advanced Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_saif

Refer to write_saif in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_scandef

Refer to write_scandef in Chapter 13, “Design for Test.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_script

```
write_script [-hdl]
     [-analyze_all_scan_chains [-dont_overlay_segments]]
     [design] [> file]
```

Generates a script that contains the timing for all modes and the design rule constraints of the design. If you used DFT functionality, the script will also contain any test constraints that were applied, as well as any objects that were created in the design as a result of inserting DFT logic such as boundary scan, building the fullscan chains, and inserting scan chain compression logic.

The resulting script can subsequently be used to examine the current design constraints, or it can be read back into Genus to perform analysis or optimization at a later time.

The `write_script` command can also compress the output using the gzip (`.gz` extension).

The script contains the following:

- The attributes connected with the `wire_load` models
- Clock objects and their reference to the pins of the design blocks
- `External_delay` on all inputs and outputs
- Timing exceptions
- `max_fanout` / `max_capacitance` and similar design rule constraints applied
- All user defined attributes that were created with the `define_attribute` command

The script can also include DFT constraints or commands, such as:

- DFT constraints created (or tool inferred) using any of the following: `define_dft shift_enable, define_dft test_mode, define_dft test_clock, define_dft scan_chain`
- DFT constraints created with `set_attribute dft_dont_scan`
- DFT objects created by the user or by the tool with `set_attribute user_defined` and `set_attribute dft_auto_created`
- `check_dft_rules`

Note: The `write` command writes out only the design itself while the `write_script` command writes out the constraints for the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-analyze_all_scan_chains`

Writes out all chains in the `dft/report/actual_scan_chains` directory using the following notation:

```
define_dft scan_chain -name name... -sdo sdo -analyze
```

When running the script in a new Genus session, the Genus-DFT engine analyzes the existing scan chains (traces the connectivity of the chains) and restores this information into the `dft/report/actual_scan_chains` directory.

`design` Specifies the name of the design for which to write a script.

`-dont_overlay_segments`

Adds the `-dont_overlay` option to the scan chains it is writing out with the `-analyze` option.

Note: You can only specify this option when you specify the `-analyze_all_scan_chains` option.

`file` Specifies the name of the file to which to write the constraints.

`-hdl` Writes out the architecture/entity filename information to the output file.

Examples

- The following example saves the design and its constraints:

```
legacy_genus:/> write_hdl > mapped.v
legacy_genus:/> write_script > mapped.g
```

The design and script is subsequently read into another Genus session. You must specify any .lib, LEF, or cap table files: these files are process specific as opposed to design specific and therefore are not automatically loaded.

```
legacy_genus:/> set_attribute library areid.lib
legacy_genus:/> set_attribute lef_library areid.lef
legacy_genus:/> set_attribute cap_table_file areid.cap
legacy_genus:/> read mapped.v
legacy_genus:/> elaborate
legacy_genus:/> source mapped.g
```

- The following example automatically compresses the output file using the `.gz` extension:

```
legacy_genus:/> write_script > foo.g.gz
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands:

create_mode on page 453
define_clock on page 456
define_cost_group on page 461
define_dft scan_chain on page 966
external_delay on page 464
multi_cycle on page 471
path_adjust on page 475
path_delay on page 479
path_disable on page 482
path_group on page 485

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sdc

```
write_sdc
     [-version {1.1|1.3|1.4|1.5|1.7|2.0}][-dc]
     [-strict] [-no_split] [-exclude cmd_list]
     [-mode mode_name] [-view analysis_view]
     [-constraint_mode constraint_mode] [design] [> file]
```

Writes out the current design constraints in Synopsys Design Constraint (SDC) format. The `write_sdc` command can also compress the SDC constraints with gzip (`.gz` extension).

When using the `write_sdc` command, Genus replaces the / character with the @ character when the `/` character is used in the name of objects. This could happen when the design is ungrouped or when the / character is used as the `ungroup_separator`.
To prevent this problem, write out the constraints using an SDC version less than 1.3 to avoid the hsc specification. For example: `legacy_genus:/> write_sdc -version 1.1`.

For those SDCs that are not supported, Genus will issue a warning message but store them for output for the `write_sdc` command. Genus will only store the SDCs and not manipulate any data with them.

Note: Using the `write_sdc` command may not capture all the design information necessary to recreate the image of a design's constraints.

Options and Arguments

`-constraint_mode` *`constraint_mode`*

Specifies the name of the constraint mode that the SDC constraints apply to.

A constraint mode groups a set of constraints to be used with a specific analysis view.

The constraint mode must have been created with the `create_constraint_mode` command.

Note: This option is not supported in the legacy UI. If you specify this option, the tool will return an error message.

`-dc`

Specfies to write the constrains in DC language instead of TCL.

`design`

Specifies the name of the design for which to write the SDC constraints.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

file	Specifies the name of the file to which to write the SDC constraints.
-exclude *cmd_list*	Specifies the commands that must not be written to the SDC file.
-mode *mode*	Writes out mode specific constraints for a design.
-no_split	Prevents printing the SDC commands over several lines.
-strict	Writes out commands that are specifically listed in the SDC specification. If you do not use this option, the `write_sdc` command outputs commands that are DC and PT compatible, which means that commands not listed in the SDC specification may be written out. See Examples for the difference in results when using the `-strict` option.
-version {1.1|1.3|1.4|1.5|1.7|2.0}	Specifies the SDC version to use. *Default*: `1.7`
-view *analysis_view*	Specifies the analysis view, the SDC constraints apply to. **Note:** This option is not supported in the legacy UI. If you specify this option, the tool will return an error message.

Examples

- The following example writes out the SDC constraints to the `my_des.sdc` file:

```
legacy_genus:/> write_sdc /designs/my_des > my_des.sdc
```

- The following example shows the results you may get if you do not specify the `-strict` option with the `write_sdc` command.

```
#######################################################################
 ...
#######################################################################

set sdc_version 1.4
# Set the current design
current_design add

set_wire_load_mode "enclosed"
set_wire_load_selection_group "ALUMINUM" -library "tutorial"
set_dont_touch [get_designs Madd_addinc]
set_dont_touch [get_cells flop1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows the results you may get if you do specify the `-strict` option with the `write_sdc` command.

```
#####################################################################
...
#####################################################################

set sdc_version 1.4

# Set the current design
current_design add

set_wire_load_mode "enclosed"
set_wire_load_selection_group "ALUMINUM" -library "tutorial"
```

- The following examples show the effect of the `-no_split` option.

  Assume that the `write_sdc` command would write out the following command in the SDC file:

```
set_false_path -from [list \
    [get_clocks in1]  \
    [get_clocks in2] ] -to [get_clocks in3]
```

  If you specify the `write_sdc` command with the `-no_split` option, the SDC command would be written as:

```
set_false_path -from [list [get_clocks in1] [get_clocks in2] ] -to [get_clocks in3]
```

- The following example illustrates the use of the `-exclude` option.

```
legacy_genus:/> write_sdc
# #####################################################################

#  Created by Genus Synthesis Solution ....

# #####################################################################

set sdc_version 1.7

set_units -capacitance 1000.0fF
set_units -time 1000.0ps

# Set the current design
current_design top

set_case_analysis 0 [get_ports in1]
create_clock -name "abc" -add -period 10.0 -waveform {0.0 5.0} [get_ports clk]
create_clock -name "abc1" -add -period 15.0 -waveform {0.0 7.5} [get_ports clk]
set_clock_gating_check -setup 0.0
set_input_delay -clock [get_clocks abc] -add_delay 0.3 [get_ports in]
set_output_delay -clock [get_clocks abc] -add_delay 0.3 [get_ports out]
set_wire_load_mode "enclosed"
set_wire_load_selection_group "ALUMINUM" -library "tutorial"
set_clock_latency  3.1 [get_clocks abc]
set_clock_uncertainty -setup 45.0 [get_clocks abc]
set_clock_uncertainty -hold 45.0 [get_clocks abc]
set_max_time_borrow 3.0 [get_clocks abc]
set_clock_latency  4.0 [get_clocks abc1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> write_sdc -exclude "set_clock_latency set_clock_uncertainty \
set_case_analysis create_clock"
```

or

```
legacy_genus:/> write_sdc -exclude {set_clock_latency set_clock_uncertainty \
set_case_analysis create_clock}
```

```
# ####################################################################

#  Created by Genus Synthesis Solution ....

# ####################################################################

set sdc_version 1.7

set_units -capacitance 1000.0fF
set_units -time 1000.0ps

# Set the current design
current_design top

set_clock_gating_check -setup 0.0
set_input_delay -clock [get_clocks abc] -add_delay 0.3 [get_ports in]
set_output_delay -clock [get_clocks abc] -add_delay 0.3 [get_ports out]
set_wire_load_mode "enclosed"
set_wire_load_selection_group "ALUMINUM" -library "tutorial"
set_max_time_borrow 3.0 [get_clocks abc]
```

Related Information

Affected by this command: create_mode on page 453

read_sdc on page 352

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sdf

```
write_sdf [-version {OVI 3.0 | OVI 2.1}]
     [-precision non_negative_integer]
     [-timescale {ps | ns}] [-delimiter character]
     [-celltiming {all | none | nochecks}]
     [-interconn {port | interconnect [-no_empty_cells]}]
     [-edges {edged | check_edge}] [-condelse]
     [-nonegchecks] [-no_escape] [-nosplit_timing_check]
     [-no_input_port_nets] [-no_output_port_nets]
     [-recrem [merge_always | merge_when_paired | split]
     [-setup [merge_always | merge_when_paired | split]
     [-design] [> file}
```

The command generates a Standard Delay Format (SDF) file that analysis and verification tools or timing simulation tools can use for delay annotation. The SDF file specifies the delay of all the cells and interconnects in the design in the Standard Delay Format. Specifically, it includes the delay values for all the timing arcs of a given cell in the design.

The `write_sdf` command can generate both plain-text and GNU zipped (.gz) SDF files.

By default, the `write_sdf` command uses the delays already cached by the system and writes only values of the triplets.

Note: Use the `write_sdf` command after technology mapping (after the `syn_map` command).

Options and Arguments

`-celltiming {all | none | nochecks}`

Specifies which cells delays and timing checks to write out.

`all`—Writes all cell delays and timing checks to the SDF file.

`none`—Prevents cell delays and timing checks from being written into the SDF file.

`nochecks`—Disables the writing of timing checks only.

Default: `all`

`-condelse`

Writes `CONDELSE` constructs with the default value when a `COND` construct is written.

Note: Use this argument with the `-version` 3.0 option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-delimiter character`

Specifies the hierarchy divider character to be used in the SDF file. The valid options are the "`/`" and "`.`" characters.

This option sets the hierarchy divider of only the `write_sdf` command. If omitted, the default hierarchy divider character (.) is used as the `write_sdf` default.

`-design`

Specifies the design name for which the SDF file has to be generated.

`-edges {edged | check_edge}`

Specifies whether to add edge specifiers on combinational arcs.

`check_edge`—Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs.

`edged`—Keeps edge specifiers on timing check arcs as well as combinational arcs.

Default: `edged`

`file`

Specifies the name of the SDF output file. If omitted, this argument defaults to stdout.

`-interconn {port | interconnect}`

Specifies the construct to use for writing out net delays.

`port`—Writes out the net delays using the `PORT` construct.

`interconnect`—Writes out the net delays using the `INTERCONNECT` construct.

Default: `port`

`-no_escape`

Writes out object names without escaping the special characters such as "`[`" or "`]`".

`-nonegchecks`

Converts all negative timing check values to `0.0`.

`-no_empty_cells`

Suppresses writing out empty cell descriptions.

`-no_input_port_nets`

Suppresses writing out nets connected to input ports.

`-no_output_port_nets`

Suppresses writing out nets connected to output ports.

`-nosplit_timing_check`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Does not split the `TIMINGCHECK` delays (`SETUP/HOLD/RECOVERY/REMOVAL` delays). Instead, the maximum delay values are used.

`-precision` *`non_negative_integer`*

Specifies the number of digits appearing after the decimal point in the output SDF file.

`-recrem [merge_always | merge_when_paired | split]`

Determines how `RECOVERY` and `REMOVAL` timing checks are written.

`merge_always`—Always writes combined `RECREM` checks, even if either of the `RECOVERY` or `REMOVAL` check does not exist.

For example, if only a `RECOVERY` check exists, the tool writes a combined `RECREM` check in the SDF file as follows:

```
RECREM (value) (value) () ()
```

`split`—Splits the `RECREM` checks into a `RECOVERY` check and a `REMOVAL` check.

`merge_when_paired`—Writes combined `RECREM` checks only when both `RECOVERY` and `REMOVAL` checks exist.

For example, if both a `RECOVERY` and a `REMOVAL` check exist, the tool writes a combined `RECREM` check in the SDF file as follows:

```
RECREM (value) (value) (value) (value)
```

However, if only the `RECOVERY` check exists, the tool only writes the `RECOVERY` check in the SDF file:

```
RECOVERY (value) (value)
```

Default: `merge_always`

`-setuphold [merge_always | merge_when_paired | split]`

Determines how `SETUP` and `HOLD` timing checks are written.

`merge_always`—Always writes combined `SETUPHOLD` checks, even if either of the `SETUP` or `HOLD` check does not exist.

For example, if only a `SETUP` check exists, the tool writes a combined `SETUPHOLD` check in the SDF file as follows:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
SETUPHOLD (value) (value) () ()
```

`split`—Splits the `SETUPHOLD` checks into a `SETUP` check and a `HOLD` check.

`merge_when_paired`—Writes combined `SETUPHOLD` checks only when both `SETUP` and `HOLD` checks exist.

For example, if both a `SETUP` and a `HOLD` check exist, the tool writes a combined `SETUPHOLD` check in the SDF file as follows:

```
SETUPHOLD (value) (value) (value) (value)
```

However, if only the `SETUP` check exists, the tool only writes the `SETUP` check in the SDF file:

```
SETUP (value) (value)
```

Default: `merge_always`

`-timescale {ps | ns}`

Specifies a multiplier to the delay values used in the SDF file. This option does not change the timescale setting of the SDF file. The delay values are scaled, but the units are the same.

Default: `ps`

`-version {OVI 3.0 | OVI 2.1}`

Specifies whether to generate SDF version 2.1 or 3.0.

Default: OVI 3.0

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Table 7-1 Definition of -`edges` Option Values

	edged value	library value	noedge value
`IOPATH from a unate input pin to output pin`	No edge specifier	No edge specifier	No edge specifier
`IOPATH from a non-unate input pin to output pin`	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either “both_edges” or “start_edge”	No edge specifier when merging edged paths, the maximum values are used for MAX and TYP fields; minimum values are used for MIN fields.
`IOPATH of a clock/enable pin to data out pin`	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either “both_edges” or “start_edge”	No edge specifier

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`Timing checks`	Edge specifiers match the check arcs specified in the timing library	Overridden by value of the `sdf_edges` attribute. For `both_edges`, the timing checks will contain edge specifier on both starting and ending pins. For `start_edge`, the starting pin (for HOLD type check) or ending pin (for SETUP type check) is edge-specified. For `end_edge`, the ending pin (for HOLD style check) or the starting pin (for SETUP type check) is edge-specified. The timing checks are duplicated or combined appropriately. The `start_edge` and `end_edge` are interpreted as first and second events, respectively, of the Verilog-XL's $hold() and $setup system tasks.	Only the edge specifiers of the reference pin are written.

Examples

- The following example writes out the SDF file, `areid.sdf`, with the "/" delimiting character:

```
legacy_genus:/> syn_map
legacy_genus:/> write_sdf -delimiter "/" > areid.sdf
```

- The following report illustrates two `TIMINGCHECK examples: the first only writes out the maximum delays while the second writes out all the delays.`

```
(TIMINGCHECK
(HOLD D (posedge CK) (::0.0))
(SETUP D (posedge CK) (::0.452))
)
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
(TIMINGCHECK

(HOLD (negedge D) (posedge CK) (::0.0))
(HOLD (posedge D) (posedge CK) (::0.0))
(SETUP (negedge D) (posedge CK) (::0.452))
(SETUP (posedge D) (posedge CK) (::0.181))
)
```

To write out only the maximum `TIMINGCHECK` delays (the first report above), use the `-nosplit_timing_check` option:

```
legacy_genus:/> write_sdf -nosplit_timing_check > areid.sdf
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sdp_file

See write_sdp_file in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_set_load

`write_set_load [design] [> file]`

Generates a set of load values, which were obtained from the physical layout estimator (PLE) or wire-load model, for all the nets in the specified design. This command is useful when performing timing correlation across various synthesis and timing tools. The `write_set_load` command can be used to reproduce PLE based timing in external timing analyzers.

Options and Arguments

design	Generates the load values for the specified design.
file	Specifies the file to which to write the load values.

Example

- The following example shows that the load values on all the nets is .0103:

```
legacy_genus:/> write_set_load
set_load  0.0103 [get_nets inst1/out1[3]]
set_load  0.0103 [get_nets inst1/out1[2]]
set_load  0.0103 [get_nets inst1/out1[1]]
set_load  0.0103 [get_nets inst1/out1[0]]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_snapshot

```
write_snapshot -directory string -tag string
     [-innovus] [-hierarchical]
     [-include_text_files] [-split_db]
     [-no_gzip] [-levels_of_logic] [design]
```

Generates design reports and database snapshots.

Options and Arguments

design	Specifies the design name on which to execute the `write_snapshot` command.
`-directory` *string*	Specifies the path to the directory where the output data will be stored.
`-hierarchical`	Writes out additional information for the hierarchical flow using interface logic models (ILMs).
`-include_text_files`	Specifies to write the design data as text as well. This is automatically enabled with the `-innovus` option.
`-innovus`	Specifies to write data for Innovus.
`-levels_of_logic`	Prints the number of gates on the critical path per cost group in the QoR report.
`-no_gzip`	Prevents that the generated files are compressed in gzip format.
`-split_db`	Writes out the setup information in a setup script and the netlist information into a setup-free database.
`-tag` *string*	Specifies a tag name for reports and design data generated at this stage.

Example

- The following example writes out the basic output of the command:

```
write_snapshot -dir snapshot -tag counter.elab
This summary table is written to the logfile:
QoS Summary for counter
==============================================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Metric                          counter.elab
================================================================================
Slack (ps):                    -714
  R2R (ps):                     606
  I2R (ps):                    -568
  R2O (ps):                    -714
  I2O (ps):                no_value
  CG  (ps):                no_value
TNS (ps):                   8,594
  R2R (ps):                     0
  I2R (ps):                 5,111
  R2O (ps):                 3,483
  I2O (ps):                no_value
  CG  (ps):                no_value
Failing Paths:                 19
Cell Area:                 3,871
Total Cell Area:           3,871
Leaf Instances:              122
Total Instances:             122
Utilization (%):           73.70
Tot. Net Length (um):    no_value
Avg. Net Length (um):    no_value
Route Overflow H (%):       0.00
Route Overflow V (%):       0.00
MBCI(%) (bits/gate) :       0.00
Norm Cong Hotspot Area:
  Max Cong:              no_value
  Tot Cong:              no_value
================================================================================
CPU  Runtime (h:m:s):    00:00:10
Real Runtime (h:m:s):    00:01:57
CPU  Elapsed (h:m:s):    00:00:16
Real Elapsed (h:m:s):    00:02:00
Memory (MB):               581.61
================================================================================
================================================================================
Flow Settings:
================================================================================
Total Runtime (h:m:s):   00:01:58
Total Memory (MB):         582.61
Executable Version:       version
================================================================================
Total Cell Area =  Cell Area + Physical Cell Area
Total Instances =  Leaf Instances + Physical Instances

The following files are written to the snapshot directory:
counter.elab_area.rpt
counter.elab_counter.db
counter.elab_gates.rpt
counter.elab_qor.rpt
counter.elab_time.rpt
final.rpt
```

Related Information

Related command: write_db on page 364

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_spef

See write_spef in Chapter 12, “Physical.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sv_wrapper

```
write_sv_wrapper
     [-allow_error]
     [-module_suffix string] [-subdesign_suffix string]
     [-exclude_type_definition]
     [-match_top_name_to_design_name]
     [-update_design_name] [design | subdesign] [> file]
```

Writes out a wrapper SystemVerilog module for a design or subdesign. Such a wrapper module can help in a comparative simulation of the output netlist from Genus and the input RTL design, especially when the design description in the input RTL has complex ports.

When you synthesize a design `test` that has complex ports in the input RTL description, the `write_sv_wrapper` command writes out the definitions of the complex port types, as well as a System Verilog description of module `test` which has the same complex ports as the original RTL description of `test`. The module's test body consists of a single instantiation of another module named `test%s` where %s is the suffix specified with the `-subdesign_suffix` option.

`write_sv_wrapper` is also used in Bottom up synthesis flow. If a sub-block has complex ports (e.g structure port or multidimensional array port), then in its pre-synthesized netlist, these ports may be represented by multiple ports, one each for each basic leaf field of the complex port. Synthesis process needs to be able to match each complex port name in instantiation with the corresponding multiple ports in the pre-synthesized netlist. A wrapper SV module is included in the netlist with the same name as the sub-block, its body consists of one instance of the pre-synthesized sub-block module using the modified name. This is done through the `write_sv_wrapper` command.

The ports of the wrapper SV module match, in type and complexity with the original definition of the sub-block in RTL.

Note: Wrapper is generated only for System Verilog design with ports of type struct, array, or interface.

Options and Arguments

`-allow_error`
: Prevents that a blocking error code is returned even if an error occurred during wrapper creation.

{*design*|*subdesign*}
: Writes out the wrapper SystemVerilog module for the specified design or subdesign.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-exclude_type_definition`

Prevents to write out complex types (structures, unions, interfaces, and so on) for the wrapper module.

If SV wrappers are written out with included type definitions (which is the default nature of `write_sv_wrapper`), Genus complain of re-declaration of type definitions since both SV wrappers and top design contain the same type definitions.

file

Specifies the file to which to write the wrapper SystemVerilog module.

`-match_top_name_to_design_name`

Specifies to match the top module name to the current design name.

`-module_suffix` *string*

Specifies the suffix to append to the module name of the wrapper module.

`-subdesign_suffix` *string*

Specifies the suffix to append to the name of the instantiated module.

`-update_design_name`

Renames the main design module and uses the value of the `-subdesign_suffix` option as suffix to the design name.

Example

Consider the following Verilog RTL input in test.v.

```
// test.v
typedef struct {
    logic x;
    logic y;
} pair;

module test( input pair in1, output out1);
    assign out1 = in1.x & in1.y;
endmodule
```

Assume the Verilog input file is synthesized through the script test.g:

```
## test.g
set_attr library tutorial.lib
read_hdl -sv test.v
elaborate
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
syn_map
write_hdl > test.vc
```

The output below shows that the complex port `in1` has been broken up during synthesis into two ports `\in1[x]` and `\in1[y]`, so that the netlist is a structural Verilog netlist. This mismatch between the ports of the module in the output and the ports of the input RTL module can create problems for simulation-based comparison between the input and output design.

```
// test.vc : the file output by RC
module test(\in1[x] , \in1[y] , out1);
    input \in1[x] , \in1[y] ;
    output out1;
    wire \in1[x] , \in1[y] ;
    wire out1;
    wire n_0;
    inv1 g8(.A (n_0), .Y (out1));
    nand2 g9(.A (\in1[x] ), .B (\in1[y] ), .Y (n_0));
endmodule
```

To avoid this port mismatch, the script should be altered as shown below:

```
## test.g
set_attr library tutorial.lib
read_hdl -sv test.v
set_attr hdl_sv_module_wrapper true /
elaborate
syn_map
write_sv_wrapper -update_design_name -subdesign_suffix _core test > test_wrap.v
write_hdl > test.vc
```

This results in the following output:

```
// File test_wrap.v

typedef struct {
    logic x;
    logic y;
} pair;

module test(
    input pair  in1,
    output logic  out1);

    test_core u1(
        .out1(out1),
        .\in1[x] (in1.x),
        .\in1[y] (in1.y));
endmodule
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
// File test.vc
module test_core(\in1[x] , \in1[y] , out1);
    input \in1[x] , \in1[y] ;
    output out1;
    wire \in1[x] , \in1[y] ;
    wire out1;
    wire n_0;
    inv1 g8(.A (n_0), .Y (out1));
    nand2 g9(.A (\in1[x] ), .B (\in1[y] ), .Y (n_0));
endmodule
```

By combining the two files (`test_wrap.v` and `test.vc`), you get a modified design hierarchy with the benefit that the top module has the same ports as the original RTL definition of the top module.

Note: The wrapper contains only the definitions of those complex types (structure, interface, union, enum, and so on) that occur in the portlist of the design, and not in its subdesigns.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_tcf

Refer to write_tcf in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_template

```
write_template
     [-dft] [-power] [-cpf] [-retime]
     [-physical] [-performance]
     [-area] [-no_sdc] [-n2n] [-yield]
     [-full] [-simple] [-split]
     [-multimode] -outfile file
```

Generates a user-friendly template script with the commands and attributes needed to run Genus. Use the command options to include specific commands and attributes in the script.

Options and Arguments

`-area`	Writes out a template script for area-critical designs.
`-cpf`	Writes out a template script for the Common Power Format (CPF) based flow and a template CPF file (`template.cpf`). The template CPF file is read in the template script with the `read_cpf` command.
`-dft`	Writes out a template script for the test synthesis (DFT) flow. The template contains commands and attributes needed for basic and advanced DFT features.
`-full`	Writes out DFT, power, and retiming commands and attributes along with the basic template.
`-multimode`	Writes out a template script for multi-mode analysis.
`-n2n`	Writes out the template script for netlist to netlist optimization in Genus. Use with the `-dft` and `-power` options to include the DFT and power attributes and commands.
`-no_sdc`	Writes out clock delays and input and output delays in the Genus format using the `define_clock` and the `external_delay` commands.
`-outfile` *`string`*	Specifies the name of the file to which the template script is to be written.
`-performance`	Writes out a template script which enables some advanced optimization algorithms that can improve QoR and that have an impact on the runtime.
`-physical`	Writes out a template script for the physical flow.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-power`	Writes out power attributes and commands.
`-retime`	Writes out retiming attributes and commands.
`-simple`	Writes out a simple template script. You cannot use this option with the `-split`, `-area`, `-dft`, `-cpf`, `-multimode`, `-physical`, `-power`, `-retime`, or `-full` options.
`-split`	Writes out a template script with a separate setup file which contains the root attributes and setup variables. If you specified the `-dft` option with the `-split` option, an additional file is created which contains the DFT design attributes, test clock and scan chain information. If you specified the `-power` option with the `-split` option, an additional file is created which contains the leakage and dynamic power, and clock-gating setup information.
`-yield`	Writes out a template script for yield.

Examples

- The following example writes out the basic template file with the constraints in SDC format:

  ```
  legacy_genus:/> write_template -outfile template.g
  ```

- The following example writes out the basic template file with the constraints in the Genus format using the `define_clock` and the `external_delay` commands:

  ```
  legacy_genus:/> write_template -no_sdc -outfile template.g
  ```

- The following example adds both the DFT and power related attributes to the template.g file written out:

  ```
  legacy_genus:/> write_template -dft -power -outfile template.g
  ```

- The following example writes out the template script with the DFT attributes and commands for netlist to netlist optimization:

  ```
  legacy_genus:/> write_template -dft -n2n -outfile template.g
  ```

- The following example writes out the template script *`template.g`* and the setup file *`setup_template.g`* that contains all the root attributes and setup variables and includes it in the *`template.g`* file:

  ```
  legacy_genus:/> write_template -split -outfile template.g
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example writes out the *template.g* template script, a setup_*template.g* setup file, a dft_*template.g* file, and a power_*template.g* file and includes them in the appropriate *template.g* file:

  ```
  write_template -split -dft -power -outfile template.g
  ```

- The following example writes out a simple template script with no path and cost groups and without any variables, power, or DFT related attributes:

  ```
  write_template -simple -outfile template.g
  ```

- The following example writes out a template script for area critical designs:

  ```
  write_template -area -outfile template.g
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_tempus

```
write_tempus [-default] [-ocv] [-pre_lib_include string]
     [-pre_include string] [-post_include string]
     [-netlist string]
     [-sdc string] [-sdf string] [-spef string]
     [> file]
```

Generates a Tempus or Encounter® Timing System (ETS) run script.

Options and Arguments

`-default`	Generates a simple ETS run script. The script will contain the following ETS commands: `read_lib`, `read_verilog`, `set_top_module`, `read_sdc`, and `report_timing`.
`file`	Redirects the output to the specified file.
`-netlist` *`string`*	Specifies the UNIX path of the file containing the gate-level netlist.
`-ocv`	Adds an extra `set_timing_derate` command into the ETS run script. This option can only be specified with the `-sdf` option.
`-post_include` *`string`*	Specifies the UNIX path of the include file that contains ETS commands that need to be added after the `report_timing` command in the run script generated by `write_ets`.
`-pre_include` *`string`*	Specifies the UNIX path of the include file that contains ETS commands that need to be added before the `report_timing` command in the run script generated by `write_ets`.
`-pre_lib_include` *`string`*	Specifies the UNIX path of the include file that contains ETS commands that need to be added before the libraries are read in the run script generated by `write_ets`.
`-sdc` *`string`*	Specifies the UNIX path of the SDC file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-sdf` *`string`*	Specifies the UNIX path of the SDF file. If this option is specified, the `read_sdf`, `set_analysis_mode`, and `set_op_cond` commands will be added to the ETS run script.
`-spef` *`string`*	Specifies the UNIX path of the SPEF file. If this option is specified, the `read_spef`, `set_analysis_mode`, and `set_op_cond` commands will be added to the ETS run script.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

8

Constraints

- clock_uncertainty on page 450
- create_mode on page 453
- define_clock on page 456
- define_cost_group on page 461
- derive_environment on page 462
- external_delay on page 464
- generate_constraints on page 468
- get_interactive_constraint_modes on page 470
- multi_cycle on page 471
- path_adjust on page 475
- path_delay on page 479
- path_disable on page 482
- path_group on page 485
- phys_enable_ocv on page 488
- phys_reset_ocv on page 489
- propagate_constraints on page 490
- set_design_mode on page 492
- set_interactive_constraint_modes on page 493
- specify_paths on page 494
- validate_constraints on page 500

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_uncertainty

```
clock_uncertainty
     [-fall | -rise]
     [-from_edge string] [-to_edge string]
     [-from clock_list | -fall_from clock_list |
       -rise_from clock_list]
     [-to clock_list | -fall_to clock_list |
       -rise_to clock_list]
     [-hold | -setup]
     [-clock clock_list] uncertainty
     [clock|port|instance|pin]...
```

Specifies the uncertainty on the clock network. You specify either a simple or an inter-clock uncertainty.

- Simple uncertainties are defined directly on a clock, port, pin, or instance.These uncertainty values are stored in attributes.
- The inter-clock uncertainties (defined using options such as `-from`, `-to`, `-rise_from`, `-rise_to`) are modeled as `path_adjust` exceptions. These uncertainties take precedence over the simple uncertainty values.

Options and Arguments

`{clock | pin | port | instance}`

Specifies the clocks, pins, ports and instances to which the specified uncertainty value applies. In case of instances, the uncertainty is applied on the input pins of the instance.

Note: These arguments only apply to simple uncertainties.

`-clock clock_list`

Specifies the clock(s) to which the uncertainty value applies.

If this option is not specified, it applies to all clocks propagating to that pin or port.

Note: This option only applies to simple uncertainties.

`-fall | -rise`

Specifies to apply the uncertainty to the falling or rising edge of the capture clock pin.

If neither option is specified, the uncertainty value applies to both edges.

Note: These options only apply to inter-clock uncertainties.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-from` *`clock_list`* | `-fall_from` *`clock_list`* | `-rise_from` *`clock_list`*

Applies the uncertainty value to the specified list of launching clocks.

Note: These options only apply to inter-clock uncertainties.

`-from_edge {rise | fall | both}`

Specifies the edge type of the launch clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_from` or `-rise_from`.

`-hold | -setup`

Specifies that the clock uncertainty applies to hold or setup checks.

If neither option is specified, the uncertainty value applies to both checks.

Note: These options apply to both types of uncertainties.

`-to` *`clock_list`* | `-fall_to` *`clock_list`* | `-rise_to` *`clock_list`*

Applies the uncertainty value to the specified list of capture clocks.

Note: These options only apply to inter-clock uncertainties.

`-to_edge {rise | fall | both}`

Specifies the edge type of the capture clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_to` or `-rise_to`.

`uncertainty`

Specifies the uncertainty value for the clock(s). Use a floating value.

Examples

- The following command sets a (simple) setup uncertainty of 0.4 sdc units for all paths ending at `reg1` and captured by the rising transition of all clocks propagating to `reg1/CK`.

```
clock_uncertainty -setup -rise 0.4 [find / -pin reg1/CK]
```

- The following command sets a (simple) setup uncertainty of 0.4 sdc units for all paths ending at `reg1` and captured by the rising transition of `clk1`.

```
clock_uncertainty -setup -rise -clock clk1 0.4 [find / -pin reg1/CK]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command sets a (inter clock) setup uncertainty of 0.5 sdc units for all paths launched by `clk2` and captured by `clk1`.

```
clock_uncertainty -setup -from clk2 -to clk1 0.5
```

Related Information

Affects this command:	path_adjust on page 475
Related command:	set_clock_uncertainty on page 1548
Sets these attributes:	clock_hold_uncertainty
	clock_setup_uncertainty
	hold_uncertainty_by_clock
	setup_uncertainty_by_clock

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_mode

```
create_mode -name mode_list
     [-default] [-design design]
```

Specifies the mode for power and timing analysis and optimization. Use this command after loading and elaborating the design, before reading in an SDC file, and before using any of the following constraints: define_clock, external_delay, multi_cycle, path_adjust, path_delay, path_disable, path_group, and specify_paths.

After creating modes, use the `-mode` option with the `read_sdc` command.

The command returns the directory path to the `mode` object that it creates. You can find the objects created by the `create_mode` command in:

`/designs/design/modes`

Options and Arguments

`-default`	Designates the specified mode as the default mode. In this case, you can specify only mode with the `-name` option.
`-design design`	Specifies the name of the design for which you want to create a mode.
`-name mode_list`	Specifies the names of the modes to be created. Unless a mode is assigned to be the default mode using the `-default` option, the first mode specified will be the default.

Examples

- The following example creates multiple modes for one design using one `create_mode` command:

  ```
  legacy_genus:/>create_mode -name "mode1 mode2" -design design_name
  ```

- The following example creates two modes using multiple `create_mode` commands, and declares one of them as the default mode.

  ```
  legacy_genus:/> create_mode -name a
  /designs/design_name/modes/a
  legacy_genus:/>create_mode -name b -default
  /designs/design_name/modes/b
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example illustrates the recommended flow.

```
read_hdl test.v
elaborate
create_mode -name {a b}
read_sdc -mode a a.sdc
read_sdc -mode b b.sdc
```

Related Information

Affects these commands:	define_clock on page 456
	external_delay on page 464
	multi_cycle on page 471
	path_adjust on page 475
	path_delay on page 479
	path_disable on page 482
	path_group on page 485
	read_sdc on page 352
	report clocks on page 600
	report timing on page 709
	specify_paths on page 494
	write_sdc on page 422
Related commands:	derive_environment on page 462
	report summary on page 703
	write_encounter on page 393
	write_script on page 419
Sets this attribute:	default
Related attributes:	(instance) disabled_arcs_by_mode
	(pin/port) external_delays_by_mode
	(design/instance) latch_borrow_by_mode
	(design/instance/pin) latch_max_borrow_by_mode
	(pin/port) propagated_clocks_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

(design/pin/port/cost group) slack_by_mode

(pin/port) timing_case_computed_value_by_mode

(instance) timing_case_disabled_arcs_by_mode

(pin/port) timing_case_logic_value_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_clock

```
define_clock -name string
     -period integer [-divide_period integer]
     [-rise integer] [-divide_rise integer]
     [-fall integer] [-divide_fall integer]
     [-domain string] [-mode mode_name]
     [-design design] [pin|port]...
```

Defines a clock waveform. A clock waveform is a periodic signal with one rising edge and one falling edge per period. The command returns the directory path to the clock object that it creates.

Note: Clock waveforms that are not applied to objects in your design are referred to as "external" clocks and are only used as references for external delay values (see the external_delay command).

Options and Arguments

`-design` *design*

Specifies the name of the top module for which you want to define a clock waveform.

This option is required for external clocks when there are multiple top designs or user netlists.

`-divide_fall` *integer*

Determines together with the `-fall` option the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a fraction of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `100`

`-divide_period` *integer*

Determines together with the `-period` option the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

Default: `1`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-divide_rise` *`integer`*

Determines, together with the `-rise` option, the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a fraction of the period and is derived by dividing `-rise` by `-divide_rise`.

Default: `100`

`-domain` *`string`*

Specifies the name of the clock domain. A clock domain groups clocks that are synchronously related to each other, allowing timing analysis to be performed between these clocks. Genus only computes timing constraints between clocks in the same clock domain.

Paths between clocks in different domains are unconstrained by default. To constrain these paths, use the `path_delay` command.

Default: `domain_1`

`-fall` *`integer`*

Determines, together with the `-divide_fall` option, the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a fraction of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `50` (falling edge is halfway through the period)

`-mode` *`mode_name`*

Defines a clock waveform for a mode.

`-name` *`string`*

Specifies the name of the clock that is being defined.

Each clock object in your design must have a unique name. If you define a new clock with the same name as an existing clock, then the new clock replaces the old one.

The clock name allows you to search for the clock later (through the find command) or to recognize it in reports.

Default: Name of the first pin or port object specified

`-period` *`integer`*

Determines, together with the `-divide_period` option, the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

{*`pin`* | *`port`*}

Specifies the clock input pin or port.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can apply clock waveforms to input ports of your design, hierarchical pins, clock pins of sequential cells in your design, a combination, or to no objects at all.

`-rise` *`integer`*

Determines together with the `-divide_rise` option the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a fraction of the period and is derived by dividing `-rise` by `-divide_rise`.

Default: `0` (rising edge is at the start of the period)

Examples

- The following example defines a clock for a design with top module `alu`.

  ```
  legacy_genus:/> define_clock -period 10000 -name 100MHz -design /designs/alu
  ```

  The clock period is 10,000 picoseconds.

- The following example defines a 300 MHz clock that applies to all sequential logic within a design:

  ```
  legacy_genus:/> define_clock -period 10000 -name 300MHz -divide_period 3
  [clock_ports]
  ```

  The clock period is 10,000/3 picoseconds. This allows Genus to compute that there are exactly 3 periods of clock `300MHz` to every period of `100MHz`.

  If you had specified a period of 3,333 picoseconds for the 300 MHz clock, Genus would compute a different relationship between the clocks (3333 periods of one clock to 10000 periods of the other) and the timing analysis of the design would be different.

- The following example defines clock `100MHz` with a rising edge after 20 percent of the period and a falling edge after 80 percent:

  ```
  legacy_genus:/> define_clock -period 10000 -name 100MHz -rise 20 -fall 80
  ```

- The following example defines clock `100MHz` with the falling edge1/3 and the rising edge 2/3 of the way through the period:

  ```
  legacy_genus:/> define_clock -period 10000 -name 100MHz -rise 2 -divide_rise 3
  \
  ==> -fall 1 -divide_fall 3
  ```

  Note: The `-divide_rise` and `-divide_fall` options allow you to precisely define when the clock transitions occur. In some cases Genus needs this precise definition to compute the correct timing constraints for paths that are launched by one clock and captured by another.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following examples create a clock domain `system` and assign the original 100 MHz and 300 MHz clocks to it:

```
legacy_genus:/> define_clock -domain "system" -period 10000 -name 100MHz
legacy_genus:/> define_clock -domain "system" -period 10000 -name 300MHz  \
==> -divide_period 3 [clock_ports]
```

- The following example saves the directory path to the clock that is defined in variable `clock1`:

```
legacy_genus:/> set clock1 [define_clock -period 10000 -name 100MHz]
```

  Alternatively, the `find` command can be used to perform a search at a later time.

- The following example removes the clock whose definition you saved in variable `clock1`:

```
legacy_genus:/> rm $clock1
```

  Note: When you remove a clock object, any external delays that reference it are also removed. Timing exceptions referring to the clock object are also removed if they can't be satisfied without the clock.

- The following example searches for a clock object by name:

```
legacy_genus:/> find / -clock 100MHz
```

- The following example examines the attributes of a clock object:

```
legacy_genus:/> ls -a [find / -clock 100MHz]
```

  Note: The clock object is identified by its directory path.

Related Information

Affects these commands:

clock_ports on page 567
external_delay on page 464
multi_cycle on page 471
path_adjust on page 475
path_delay on page 479
path_disable on page 482
path_group on page 485
report clocks on page 600
report qor on page 690
specify_paths on page 494

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	read_sdc on page 352
	report_clocks on page 600
	report_summary on page 703
	report_timing on page 709
	syn_generic on page 526
	write_encounter on page 393
	write_script on page 419
	write_sdc on page 422
Related command:	create_mode on page 453
Affects these attributes:	divide_fall
	divide_period
	divide_rise
	fall
	period
	rise
	propagated_clocks_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_cost_group

```
define_cost_group -name string
     [-weight integer] [-design design]
```

Defines a cost group. The command returns the directory path to the object that it creates.

Options and Arguments

`-design` *`design`*	Specifies the name of the design for which you want to define the cost group.
`-name` *`string`*	Specifies the name of the cost group. *Default*: `grp_x`
`-weight` *`integer`*	Specifies the weight of the cost group. The weight factor is only taken into account during incremental optimization. When two paths have identical cost factors, the tool will pick the one with the highest weight. *Default*: `1`

Examples

The following example assigns a weight factor of 1 to cost group `I2O` for the top-level design.

```
legacy_genus:/> define_cost_group -name I2O -weight 1
/designs/.../timing/cost_groups/I2O
```

Related Information

Affects these commands:	path_group on page 485
	report timing on page 709
	write_script on page 419
	write_script on page 419
Sets this attribute:	weight

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

derive_environment

```
derive_environment
     [-name name] [-sdc_only] [-power_intent] instance
```

Creates a new design from the specified instance and creates timing constraints for all modes for this design based on the timing constraints that the instance had in the original design.

This command does not perform time-budgeting. The slack at each pin in the new design matches the slack at the corresponding pin in the original design.

Note: By default, the `derive_environment` command uses Genus's more powerful constraint language that produces a more accurate timing view, but is not understood by other tools. Use the `-sdc_only` option to specify that Genus only apply constraints to the new design that can be expressed in SDC.

If you use the `derive_environment` command to generate constraints for a subdesign then try to write them out, often the constraints cannot be expressed in SDC and you will get the following error message:

```
Error   : The design contains constraints which have no SDC equivalent.[SDC-19]
        : The design is /designs/Madd_addinc.
        : If the design constraints were created using the 'derive_environment'
command, use the '-sdc_only' option so that only constraints that can be expressed
in SDC are generated. By default the 'derive_environment'command uses the more
powerful RC constaints, which cannot always be converted to SDC.
```

Options and Arguments

instance	Specifies the name of the instance whose environment (constraints) you want to derive.
`-name` *string*	Specifies the name of the target design.
`-power_intent`	Derives the power intent for the specified instance. You can only use this option after you executed the `commit_power_intent` command. In addition, this option can currently only be used for non-hierarchical CPF files.d *Important* This option is only available as a limited access feature.
`-sdc_only`	Specifies that the tool only apply constraints to the new design that can be expressed in SDC.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Using this option makes the new constraints less accurate, but makes it possible to use the constraints in flows between different tools that support SDC.

By default, the `derive_environment` command uses Genus's more powerful constraint language that produces a more accurate timing view, but is not understood by other tools.

Related Information

Affected by this command:	path_adjust on page 475
	create_mode on page 453
Affects these commands:	report timing on page 709
	syn_map on page 529
Sets this attribute:	precluded_path_adjusts

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

external_delay

```
external_delay
     {-input min_rise min_fall max_rise max_fall
     |-output min_rise min_fall max_rise max_fall}
     [-clock  object] [-edge_fall | -edge_rise]
     [-level_sensitive] [-accumulate]
     [-mode mode_name] [-name string] {port|pin}...
```

Constrains ports and pins within your design. Timing is specified as either an input or output delay, and is specified relative to a clock edge.

External delays are most often specified on top-level ports of your design.

Options and Arguments

`-accumulate`
: Indicates that more than one external delay can be specified per clock per phase.

`-clock` *`object`*
: Specifies the reference clock. Input and output delays are defined relative to this clock.

  For an input delay, the reference clock is called the *launching* clock.

  For an output delay, the reference clock is called the *capturing* clock.

  The reference clock must have been defined with the `define_clock` command.

`[-edge_rise | -edge_fall]`
: Specifies to use the rising or falling edge of the reference clock as reference edge.

  Default: `-edge_rise`

`-input` *`min_rise min_fall max_rise max_fall`*
: Specifies an input delay. That is the time between the reference edge of the launching clock and the time when the input signal at the specified ports or pins becomes stable.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can specify one, two, or four integers. If you specify one value, that value is used for all four delay values. If you specify two values, the first value defines the (minimum and maximum) rise delays, while the second value defines the (minimum and maximum) fall delays.

The *minimum* rise (fall) delays are used for hold analysis.The *maximum* rise (fall) delays are used for setup analysis.

Note: Genus does not support hold analysis.

The (minimum and maximum) *rise* delays are measured with respect to the rising edge of the input signal.

The (minimum and maximum) *fall* delays are measured with respect to the falling edge of the input signal.

`-level_sensitive`

Used with the `-input` option, it specifies the constraint coming from a level-sensitive latch.

Used with the `-output` option, it specifies the constraint to a level-sensitive latch.

`-mode` *`mode_name`*

Constrains ports and pins by mode in a design.

`-name` *`string`*

Associates a name with the specified timing constraint. If another timing constraint already exists with that name, the existing timing constraint is replaced with the new one.

The name may be useful for later finding the constraint (with the find command) and for recognizing the constraint in reports.

Default: xx_*n*

`-output` *`min_rise min_fall max_rise max_fall`*

Specifies an external output delay. That is the delay between the time when the output signal at the specified ports or pins becomes stable and the reference edge of the capturing clock.

You can specify one, two, or four integers. If you specify one value, that value is used for all four delay values. If you specify two values, the first value defines the (minimum and maximum) rise delays, while the second value defines the fall delays.

The *minimum* rise (fall) delays are used for hold analysis.The *maximum* rise (fall) delays are used for setup analysis.

Note: Genus does not support hold analysis.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The (minimum and maximum) *rise* delays are measured with respect to the rising edge of the output signal.

The (minimum and maximum) *fall* delays are measured with respect to the falling edge of the output signal.

`{pin | port}` Specifies delay for timing start points and end points, which can be primary ports, pins of sequential instances, and pins of unresolved references.

Use the `break_timing_paths` attribute to make a non-start/end point a start/end point, which then can be used with the `external_delay` command.

Examples

- The following example specifies an input delay of 300 picoseconds on all bits of port `a` relative to the falling edge of clock `clock1`:

  ```
  legacy_genus:/> external_delay -input 300 -edge_fall -clock [find / -clock
  clock1] \
  [find / -port a*]
  ```

  Genus interprets this as a worst-case upper bound constraint, that is, the latest time to set up the data on a D pin of an edge-triggered flop.

  Applying delays to individual bits of multibit ports or busses is possible since each bit of a port is accessible individually within the directory structure of the design.

- The following example specifies an output delay of 1300 picoseconds on all bits of port `a` relative to the rising edge (default) of clock `clock1`:

  ```
  legacy_genus:/> external_delay -output 1300 -clock [find / -clock clock1] \
  [find / -port a*]
  ```

  Genus interprets this as a minimum setup time for external logic.

Related Information

Affected by this command:	define_clock on page 456
Affects these commands:	report qor on page 690
	report timing on page 709
	syn_map on page 529
	write_encounter on page 393

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	write_sdc on page 422
	write_script on page 419
Related commands:	create_mode on page 453
	define_clock on page 456
	derive_environment on page 462
	multi_cycle on page 471
	path_adjust on page 475
	path_delay on page 479
	path_disable on page 482
	path_group on page 485
	specify_paths on page 494
Affects these attributes:	External Delay Attributes
Related attributes:	(pin/port) external_delays_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

generate_constraints

```
generate_constraints [-rtl] [-netlist string]
     [-slack integer] [-report string]
     [-in_sdc string] [-out_sdc string]
     {-fpgen | -dfpgen | -trv} [-mode string] [> file]
```

Verifies the false paths in the SDC files against the RTL or netlist and then generates any missing functional false paths. The command can be used any time after elaboration. If you do not specify either the `-rtl`, `-netlist`, or `-in_sdc` options, Genus will internally generate the SDC constraints and verify them against the design at its current state. Genus will generate any missing constraints.

Use this command in the False Path Generation, Directed False Path Generation, and Timing Report Validation flows.

Options and Arguments

`-dfpgen`	Generates the false paths from Directed False Path Generation flow.
`file`	Specifies the name of the file to write the report.
`-fpgen`	Generates the false paths from False Path Generation flow.
`-in_sdc` *`string`*	Specifies the UNIX path to the SDC files.
`-logfile` *`string`*	Creates a separate CCD logfile. You must specify the UNIX path to the file.
`-mode` *`string`*	Generates mode-specific constraints for a design. If you omit this option and you did not specify any input SDC files, the command will generate the constraints for all the modes.
`-netlist` *`string`*	Specifies the UNIX path to the netlist.
`-out_sdc` *`string`*	Specifies the name for the Genus generated SDC file. *Default:* `cfp.sdc`
`-report` *`string`*	Specifies the name of the timing report file to be generated for the design. The file will be in the CCD format.
`-rtl`	Indicates that the verification should happen against the RTL.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-slack` *`integer`*	Specifies a slack value in picoseconds. Only paths below this slack value will be used for generating the timing report. This must be used with the `-report` option.
`-trv`	Generates the false paths from the Timing Report Validation flow.

Examples

- The following command generates the false path from the False Path Generation flow, verifies against the RTL, specifies the input SDC file `top.sdc`, and specifies the output SDC file `top_out.sdc`:

```
legacy_genus:/> generate_constraints -fpgen -rtl -in_sdc /home/test/top.sdc \
    -out_sdc /home/cody/top_out.sdc
```

- The following command generates the false path from the Directed False Path Generation flow, verifies against the netlist `generic.nl.v`, specifies the input SDC file `top.sdc`, specifies the slack (2 picoseconds), specifies the report name, and specifies the output SDC file `top_out.sdc`:

```
legacy_genus:/> generate_constraints -dfpgen -netlist generic.nl.v -in_sdc
top.sdc \
    -report top_out.rep -slack 2 -out_sdc top_out.sdc
```

Related Information

Affected by this attribute:	wccd_threshold_percentage
Related command:	validate_constraints on page 500

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_interactive_constraint_modes

```
get_interactive_constraint_modes
```

Returns a list of the MMMC constraint modes that are in *interactive* constraint entry mode.

When the software is in interactive constraint entry mode (by the set_interactive_constraint_modes on page 493), any specified timing constraints take effect immediately on all active analysis views that are associated with these constraint modes.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

multi_cycle

```
multi_cycle
     { {-from {instance|external_delay|clock|port|pin}...
       |-through {instance|port|pin}...[-through...]...
       |-to {instance|external_delay|clock|port|pin}...}...
       |-paths string}
     [-launch_shift integer] [-capture_shift integer]
     [-setup] [-hold] [-lenient]
     [-mode mode_name] [-name string]
```

Creates a timing exception object that overrides the default clock edge relationship for paths that meet the path selection criteria. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options. The command returns the directory path to the object that it creates.

Genus normally computes timing constraints for paths based on the launching clock waveforms and capturing clock waveforms. The default timing constraint is the smallest positive difference that exists between a launching clock edge and a capturing clock edge.

Options and Arguments

`-capture_shift` *`integer`*

Specifies the capture clock shift value.

An ordinary two-cycle path would be specified using `-capture_shift 2`. Incrementing the `-capture_shift` value adds a cycle to the path by shifting the capture edge one period later.

Default: `1`

`-hold`

Specifies that the exception is for hold timing analysis only.

Default: `setup`

`-from {`*`instance`*`|`*`external_delay`*`|`*`clock`*`|`*`port`*`|`*`pin`*`}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which the specified external delay timing exception applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Only paths that start at one of the ports or pins, or paths that are launched by one of the clock objects will have the timing exception applied to them.

`-launch_shift` *`integer`*

Specifies the launch clock shift value. Incrementing the `-launch_shift` value adds a cycle to the path by shifting the launch edge one period earlier.

Adjusting the launch edge is only useful if the launch and capture clocks have different periods. Otherwise an equivalent timing relationship can be achieved by shifting the capture clock instead.

Default: `0`

`-lenient`

Converts an invalid start or end point into a through point and issues a warning message. Without this option, an invalid start or end point results in an error message.

`-mode` *`mode_name`*

Creates a timing exception object for the specified mode that overrides the default clock edge relationship for paths that meet the path selection criteria.

`-name` *`string`*

Associates a name with the specified timing exception. If another timing exception already exists with that name, the existing timing exception is replaced with the new one.

The name can be useful for later finding the exception (with the find command) and for recognizing the exception in reports.

Default: `mc_n`

`-paths` *`string`*

Specifies the paths to which the exception should be applied. The string argument should be created using the `specify_paths` command. The `-paths` option cannot be used in conjunction with any of the other three path selection options (`-from`, `-through`, `-to`).

`-setup`

Specifies that the exception is for setup timing analysis only.

Default: `setup`

`-through {`*`instance`*`|`*`port`*`|`*`pin`*`}`

Specifies a Tcl list of a sequence of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {instance |external_delay|clock|port|pin}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which the specified external delay timing exception applies.

Only paths that end at one of the ports or pins, or paths that are captured by one of the clock objects, have the exception applied to them.

Examples

- The following example requires a path to first pass through object `a`, then end on object `b`:

  ```
  legacy_genus:/> multi_cycle -through a -to b
  ```

- The following example requires a path to pass through either object `a` or `b`.

  ```
  legacy_genus:/> multi_cycle -through {a b}
  ```

- The following example requires a path to first pass through object `a` or `b`, then pass through object `c` or `d`.

  ```
  multi_cycle -through {a b} -through {c d}
  ```

  The candidate paths would be:

  ```
  ac      ad      bc      bd
  ```

- The following example requires a path that goes through object `a`, `b`, and `c` in that order:

  ```
  legacy_genus:/> multi_cycle -through a -through b -through c
  ```

- The following example uses a Tcl variable to save the timing exception for future reference:

  ```
  legacy_genus:/> set two_cycle [multi_cycle -capture_shift 2 -from [find / -port
  a]]
  ```

  The following example removes the timing exception that you saved in the `two_cycle` variable:

  ```
  legacy_genus:/> rm $two_cycle
  ```

- The following example searches for a timing exception that you defined earlier:

  ```
  legacy_genus:/> multi_cycle -capture_shift 2 -from [find / -port a] -name
  two_cycle
  /designs/alu/timing/exceptions/multi_cycles/two_cycle
  ...
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
...
legacy_genus:/> find / -exception two_cycle
/designs/alu/timing/exceptions/multi_cycles/two_cycle
```

- The following example lists multi cycle timing exception objects using the `ls` command.

```
legacy_genus:/> ls -l timing/exceptions/multi_cycles
```

- The following examples are equivalent and apply to paths starting from (clock) pin `clk1`:

```
multi_cycle -from clk1
multi_cycle -paths [specify_paths -from clk1]
```

Related Information

Affects these commands:	report qor on page 690
	report timing on page 709
	specify_paths on page 494
	syn_map on page 529
	write_encounter on page 393
	write_sdc on page 422
	write_script on page 419
Related commands:	create_mode on page 453
	define_clock on page 456
	external_delay on page 464
	path_adjust on page 475
	path_delay on page 479
	path_disable on page 482
	path_group on page 485
	specify_paths on page 494
Sets these attributes:	Exception Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

path_adjust

```
path_adjust -delay integer
     { {-from {instance|external_delay|clock|port|pin}...
       |-through {instance|port|pin}...[-through...]...
       |-to {instance|external_delay|clock|port|pin}...}...
       |-paths string }
     [-lenient] [-mode mode_name] [-name string]
```

Modifies path constraints. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options.

These path constraints could have been

- Computed by the timing engine using the launching and capturing waveforms
- Set explicitly with the `path_delay` command

The command returns the virtual directory path to the object that it creates.

The constraints specified with the `path_adjust` command can co-exist with other timing exceptions, such as `path_delay`, `multi_cycle`, as well as `path_adjust` (it is possible to have multiple `path_adjust` constraints on a path).

The constraints created by the `path_adjust` commands can be found in:

`/designs/design_name/timing/exceptions/path_adjusts`

Once a path adjust is defined, all operations that require timing will be impacted by the adjust.This includes all reporting commands that report timing, all optimization commands, and all timing related attributes. For this reason, it is recommended to remove any path adjusts before reporting final timing.

A common use of path adjusts is to tighten selected paths for increased optimization, or to relax paths that are not critical. For example, early in the design, some blocks may not be complete from a timing perspective, and so may have unfeasible paths. A path adjust can be used so the optimization engines do not try to unfeasible paths. The advantage to this methodology is that the constraints remain unchanged, and the path adjust appears clearly in the timing report.

Similarly, a path adjust can be used to tight paths up for increased optimization. By applying a negative path adjust, the tool will see tighter timing on the specific paths, and will structure them differently. This technique can be used to improve synthesis runs based on previous analysis of results.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-delay` *`integer`*
: Specifies the delay constraint value, in picoseconds, by which the path has to be adjusted. A positive adjustment relaxes the clock constraint and a negative adjustment tightens it.

`-from {`*`instance`*`|`*`external_delay`*`|`*`clock`*`|`*`port`*`|`*`pin`*`}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which the specified external delay timing exception applies.

Only paths that start at one of the ports or pins, or paths that are launched by one of the clock objects will have the timing exception applied to them.

`-lenient`
: Converts an invalid start or end point into a through point and issues a warning message. Without this option, an invalid start or end point results in an error message.

`-mode` *`mode_name`*
: Modifies path constraints for the specified mode.

`-name` *`string`*
: Associates a name with the specified timing exception. If another timing exception already exists with that name, the existing timing exception is replaced with the new one.

The name may be useful for later finding the exception (with the find command) and for recognizing the exception in reports.

Default: `adj_`*`n`*

`-paths` *`string`*
: Specifies the paths to which the exception should be applied. The string argument should be created using the `specify_paths` command. The `-paths` option cannot be used in conjunction with any of the other three path selection options (`-from`, `-through`, `-to`).

`-through {`*`instance`*`|`*`port`*`|`*`pin`*`}`

Specifies a Tcl list of a sequence of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {`*`instance`* `|`*`external_delay`*`|`*`clock`*`|`*`port`*`|`*`pin`*`}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which the specified external delay timing exception applies.

Only paths that end at one of the ports or pins, or paths that are captured by one of the clock objects, have the exception applied to them.

Examples

- The following example tightens all paths being driven by $clock by 500ps:

  This shows how to set a tcl variable so it can be removed later.

  ```
  legacy_genus:/> set override [path_adjust -to $clock -delay -500]
  legacy_genus:/> puts $override
  /designs/test/timing/exceptions/path_adjusts/adj_1
  ```

  Now remove the path adjust

  ```
  legacy_genus:/> rm $override
  ```

- The following example applies a path adjust using a name:

  ```
  legacy_genus:/> path_adjust -to $clock -delay -500 -name override
  /designs/test/timing/exceptions/path_adjusts/override
  ...
  legacy_genus:/> find / -exception override
  /designs/test/timing/exceptions/path_adjusts/override
  ```

- The following examples are equivalent and apply to paths starting from (clock) pin `clk1`:

  Relax all timing on paths starting from clk1 by 250 ps:

  ```
  legacy_genus:/> path_adjust -from clk1 -delay 250
  legacy_genus:/> path_adjust -paths [specify_paths -from clk1] -delay 250
  ```

- This example shows how the timing report reports the use of path adjust:

  ```
  legacy_genus:/> path_adjust -name outDel -delay -500 [find / -port q]
  legacy_genus:/> report_timing
  Pin             Type            Fanout Load   Slew     Delay Arrival
                                         (fF)   (ps)      (ps)  (ps)
  ----------------------------------------------------------------
  (clock ck)      launch                                        0 R
  q_reg/clk                                        0             0 R
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
q_reg/q     (u)  unmapped_d_flop       1  0.4  0   +208   208 R
q           <<<  interconnect                  0     +0     208 R
                 out port                           +0     208 R
(ou_del)         ext delay                       +100     308 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock ck)       capture                                   500 R
                 adjustments                     -500        0
-----------------------------------------------------------------

Exception    : 'path_adjusts/outDel' path adjust    -500ps
Timing slack :    -308ps (TIMING VIOLATION)
Start-point  : q_reg/clk
End-point    : q
```

Related Information

Affects these commands:	report timing on page 709
	report qor on page 690
	specify_paths on page 494
	syn_map on page 529
	write_script on page 419
	write_encounter on page 393
	write_sdc on page 422
	write_script on page 419
Related commands	define_clock on page 456
	external_delay on page 464
	multi_cycle on page 471
	path_delay on page 479
	path_disable on page 482
	path_group on page 485
	specify_paths on page 494
Sets these attributes:	Exception Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

path_delay

```
path_delay -delay integer [-name string]
     { {-from {instance|external_delay|clock|port|pin}...
       |-through {instance|port|pin}...[-through...]...
       |-to {instance|external_delay|clock|port|pin}...}...
       |-paths string}
     [-lenient] [-mode mode_name]
```

Creates a timing exception object that allows you to specify the timing constraint for paths that meet the path selection criteria. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options. The command returns the directory path to the object that it creates.

Genus normally computes timing constraints for paths based on the launching clock waveforms and capturing clock waveforms. The default timing constraint is the smallest positive difference that exists between a launching clock edge and a capturing clock edge.

The constraints created by the `path_delay` commands can be found in:

`/designs/../timing/exceptions/path_delays`

Options and Arguments

`-delay` *integer*	Specifies the delay constraint value, in picoseconds, for paths that meet the path selection criteria.
`-from {`*instance* | *external_delay* | *clock* | *port* | *pin*`}`	Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which the specified external delay timing exception applies. Only paths that start at one of the ports or pins, or paths that are launched by one of the clock objects will have the timing exception applied to them.
`-lenient`	Converts an invalid start or end point into a through point and issues a warning message. Without this option, an invalid start or end point results in an error message.
`-mode` *mode_name*	Creates a timing exception object for the specified mode that lets you specify the timing constraint for paths that meet the path selection criteria.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-name` *`string`*

Associates a name with the specified timing exception. If another timing exception already exists with that name, the existing timing exception is replaced with the new one.

The name may be useful for later finding the exception (with the find command) and for recognizing the exception in reports.

Default: `del_n`

`-paths` *`string`*

Specifies the paths to which the exception should be applied. The string argument should be created using the `specify_paths` command. The `-paths` option cannot be used in conjunction with any of the other three path selection options (`-from`, `-through`, `-to`).

`-through {`*`instance`* `|` *`port`* `|` *`pin`*`}`

Specifies a Tcl list of a sequence of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {`*`instance`* `|` *`external_delay`* `|` *`clock`* `|` *`port`* `|` *`pin`*`}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which the specified external delay timing exception applies.

Only paths that end at one of the ports or pins, or paths that are captured by one of the clock objects, have the exception applied to them.

Examples

- The following example specifies a path delay of 5000 ps for all paths starting from port `a`.

```
legacy_genus:/> path_delay -delay 5000 -from [find / -port a]
```

- The following command defines path delay `my_delay` of `4000ps` for all paths ending at port `b`.

```
legacy_genus:/> path_delay -delay 4000 -to [find / -port b] -name my_delay
/designs/alu/timing/exceptions/path_delays/my_delay
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The following command searches for the timing exception `my_delay` defined earlier:

```
legacy_genus:/> find / -exception my_delay
/designs/alu/timing/exceptions/path_delays/my_delay
```

- The following examples are equivalent and define a path delay for all paths starting from (clock) pin `clk1`:

```
path_delay -from clk1
path_delay -paths [specify_paths -from clk1]
```

Related Information

Affects these commands:	report qor on page 690
	report timing on page 709
	specify_paths on page 494
	syn_map on page 529
	write_script on page 419
	write_sdc on page 422
Related commands:	create_mode on page 453
	define_clock on page 456
	external_delay on page 464
	multi_cycle on page 471
	path_adjust on page 475
	path_disable on page 482
	path_group on page 485
	specify_paths on page 494
Sets these attributes:	Exception Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

path_disable

```
path_disable [-name string]
     { {-from {instance|external_delay|clock|port|pin}...
       |-through {instance|port|pin}...[-through...]...
       |-to {instance|external_delay|clock|port|pin}...}...
       |-paths string}
     [-lenient] [-mode mode] [-setup| -hold]
```

Creates a timing exception object that allows you to unconstrain paths. The command returns the directory path to the object that it creates. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options.

Genus computes timing constraints for paths based on the launching clock waveforms and the capturing clock waveforms. The default timing constraint is the smallest positive difference that exists between a launching clock edge and a capturing clock edge.

Options and Arguments

`-from {`*`instance`* | *`external_delay`* | *`clock`* | *`port`* | *`pin`*`}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which the specified external delay timing exception applies.

Only paths that start at one of the ports or pins, or paths that are launched by one of the clock objects will have the timing exception applied to them.

`-lenient`

Converts an invalid start or end point into a through point and issues a warning message. Without this option, an invalid start or end point results in an error message.

`-mode` *`mode`*

Creates a timing exception object for the specified mode that lets you unconstrain paths.

`-name` *`string`*

Associates a name with the specified timing exception. If another timing exception already exists with that name, the existing timing exception is replaced with the new one.

The name may be useful for later finding the exception (with the find command) and for recognizing the exception in reports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: `dis_n`

`-paths string`
: Specifies the paths to which the exception should be applied. The string argument should be created using the `specify_paths` command. The `-paths` option cannot be used in conjunction with any of the other three path selection options (`-from`, `-through`, `-to`).

`[-setup|-hold]`
: Limits the exception to either setup timing analysis or hold timing analysis.

  By default, the exceptions applies to both.

`-through {instance | port | pin}`
: Specifies a Tcl list of a sequence of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

  You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {instance | external_delay | clock | port | pin}`
: Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which the specified external delay timing exception applies.

  Only paths that end at one of the ports or pins, or paths that are captured by one of the clock objects, have the exception applied to them.

Examples

- The following example uses a Tcl variable to save the timing exception for future reference:

  ```
  legacy_genus:/> set false_path [path_disable -from [find / -port a]]
  ```

- The following example removes the timing exception that you saved in variable `false_path`:

  ```
  legacy_genus:/> rm $false_path
  ```

- The following example lists timing exception objects using the `ls` command.

  ```
  legacy_genus:/> ls -l timing/exceptions/path_disables
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following examples are equivalent and apply to paths starting from (clock) pin `clk1`:

```
path_disable -from clk1
path_disable -paths [specify_paths -from clk1]
```

Related Information

Affects these commands:	report qor on page 690
	report timing on page 709
	specify_paths on page 494
	syn_map on page 529
	write_encounter on page 393
	write_script on page 419
	write_sdc on page 422
Affected by these commands:	define_clock on page 456
	multi_cycle on page 471
Related commands:	create_mode on page 453
	define_clock on page 456
	external_delay on page 464
	path_adjust on page 475
	path_delay on page 479
	path_group on page 485
	specify_paths on page 494
Sets these attributes:	Exception Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

path_group

```
path_group [-group string]
     {{-from {instance|external_delay|clock|port|pin}...
      |-through {instance|port|pin}...[-through...]...
      |-to {instance|external_delay|clock|port|pin}...}...
      |-paths string}
     [-lenient] [-mode mode_name] [-name string]
```

Assigns paths that meet the path selection criteria to a cost group. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options.

Options and Arguments

`-from {instance | external_delay | clock | port | pin}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports.

`-group string`

Specifies the name of the cost group (defined with `define_cost_group`) to which the path is added.

`-lenient`

Converts an invalid start or end point into a through point and issues a warning message. Without this option, an invalid start or end point results in an error message.

`-mode mode_name`

Assigns paths for a specified mode that meet the path selection criteria to a cost group.

`-name string`

Associates a name with the specified timing exception.

`-paths string`

Specifies the paths to which the exception should be applied. The string argument should be created using the `specify_paths` command. The `-paths` option cannot be used in conjunction with any of the other three path selection options (`-from`, `-through`, `-to`).

`-through {instance | port | pin}`

Specifies a Tcl list of a sequence of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {instance | external_delay | clock | port | pin}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports.

Examples

- The following example assigns the paths from all inputs to all outputs to the group called I2O, which was previously defined by a `define_cost_group` command:

```
path_group -from /designs/*/ports_in/* -to /designs/*/ports_out/* -group I2O
```

- The following examples are equivalent and apply to paths starting from (clock) pin `clk1`:

```
path_group -from clk1
path_group -paths [specify_paths -from clk1]
```

Related Information

Affects these commands:	define_cost_group on page 461
	report qor on page 690
	report timing on page 709
	specify_paths on page 494
	syn_map on page 529
	write_encounter on page 393
	write_script on page 419
	write_sdc on page 422
Related commands:	create_mode on page 453
	define_clock on page 456
	external_delay on page 464
	multi_cycle on page 471
	path_adjust on page 475

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	path_delay on page 479
	path_disable on page 482
	specify_paths on page 494
Sets these attributes:	Exception Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

phys_enable_ocv

```
phys_enable_ocv
     [-socv_to_aocv|-native_socv|-native_aocv]
     -design design
```

Controls enabling of the Advanced On-Chip Variation (AOCV) and Statistical On-Chip Variation (SOCV) flows.

Options and Arguments

`-design` *`design`*	Specifies the name of the design for which to enable the specified flow.
`-native_aocv`	Enables the native AOCV flow.
`-native_socv`	Enables the native SOCV flow.
`-socv_to_aocv`	Enables the SOCV to AOCV flow.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

phys_reset_ocv

```
phys_reset_ocv
     -design design
```

Resets the Advanced On-Chip Variation (AOCV) and Statistical On-Chip Variation (SOCV) flows.

Options and Arguments

`-design` *`design`*	Specifies the name of the design for which to reset the flow.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

propagate_constraints

```
propagate_constraints
     -block_sdc string
     [-glue_sdc string]
     [-partial_chip_sdc string]
     [-out_sdc string] [-netlist string]
     [-rule_instance_file string]
     [-rule_instance_template string]
     [-logfile string] [> file]
```

Propagates the block-level constraints to the top-level and integrates them to generate a chip-level constraints. Propagating and integrating of the design constraints requires to analyze the provided block-level and glue constraints, and to check them against any existing chip-level constraints to determine whether to ignore or promote them to the top-level.

Options and Arguments

`-block_sdc string`	Specifies a list of block names with their associated block-level SDC files in the following format: `{{block_name block_sdc_file}...}` **Note:** You need to specify the path to the SDC files. If the paths contain Tcl variables, use the format shown in the Examples on page 491.
file	Specifies the file to which the report must be written.
`-glue_sdc string`	Specifies the name of a glue SDC file. This file contains a set of constraints for the top-level module only (without covering any block-level constraints).
`-logfile string`	Creates a separate CCD logfile. You must specify the UNIX path to the file.
`-netlist string`	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
`-out_sdc string`	Specifies the name of the constraints file that is generated after propagation and integration of the block and glue constraints. *Default*: `chip.sdc`
`-partial_chip_sdc string`	

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the name of the partial SDC file that corresponds to the top-level of the design.

`-rule_instance_file` *`string`*

Specifies the name of the file which defines the rule instances for the Conformal ® Constraint Designer (CCD) tool.

`-rule_instance_template` *`string`*

Creates a template with the default rules. You can modify this file according to the design. To use this file as input for the Conformal ® Constraint Designer (CCD) tool, specify the file as value of the `-rule_instance_file` option.

Examples

- The following command creates a top-level SDC file, `chip.sdc`, which integrates the block-level SDC files `i1.sdc` and `i2.sdc`.

```
legacy_genus:/> propagate_constraints -block_sdc {{i1 i1.sdc} {i2 i2.sdc}}
```

The internally generated CCD dofile will be similar to:

```
read library -statetable -liberty ./tech/slow.lib
add search path -design .
read design -verilog ./ti1.v -lastmod -noelab
elaborate design
dofile ./default_rule_instances.do
read hierarchical sdc \
-sdc_design i1 i1.sdc \
-sdc_design i2 i2.sdc
set system mode verify
integrate -all ./chip.sdc -replace
report rule check
report environment
```

- The following example shows the use of Tcl variables to specify the paths to the SDC files:

```
propagate_constraints -block_sdc "{block1 $WORKINGDIR/block1.sdc} \
{block2 $WORKINGDIR/block2.sdc} ..."
or
propagate_constraints -block_sdc [list \
        [list block1 ${WORKINGDIR}/block1.sdc] \
        [list block2 ${WORKINGDIR}/block2.sdc] \] ...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_design_mode

```
set_design_mode
     [-process integer] [-reset]
```

Specifies the process technology value. You can use this command to change the process technology dependent default settings globally for each application instead of setting several mode options.

Options and Arguments

`-process` *`integer`*	Specifies the process technology value to set for all the applications. Units in nanometers (nm). You can specify a number from 7 (minimum value) to 250 (maximum value) *Default*: `90`
`-reset`	Resets parameters to their default values. The -reset parameter must be the first parameter specified.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_interactive_constraint_modes

```
set_interactive_constraint_modes
     constraint_modes
```

Specifies the mode(s) to which subsequent constraints—entered *interactively* at the Genus prompt—must be applied.

Once this command is set, all constraints that need a mode specification will apply the constraint to the specified mode.

Options and Arguments

constraint_modes	Specifies the list of constraint modes to which subsequent constraints must be applied.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

specify_paths

```
specify_paths
     [-mode mode_name] [-view view] [-domain clock_domain]
     [ -from {pin|port|clock|external_delay|instance}...
     | -from_rise_clock {pin|port|clock|external_delay|instance}...
     | -from_fall_clock {pin|port|clock|external_delay|instance}...
     | -from_rise_pin {pin|port|clock|external_delay|instance}...
     | -from_fall_pin {pin|port|clock|external_delay|instance}...]
     [ -through {pin|port|instance}...
     | -through_rise_pin {pin|port|instance}...
     | -through_fall_pin {pin|port|instance}...] ...
     [ -to {pin|port|clock|external_delay|instance}... |
     | -to_rise_clock {pin|port|clock|external_delay|instance}...
     | -to_fall_clock {pin|port|clock|external_delay|instance}...
     | -to_rise_pin {pin|port|clock|external_delay|instance}...
     | -to_fall_pin {pin|port|clock|external_delay|instance}...]
     [ -capture_clock_pins pin...
     | -capture_clock_pins_rise_clock pin...
     | -capture_clock_pins_fall_clock pin...
     | -capture_clock_pins_rise_pin pin...
     | -capture_clock_pins_fall_pin pin... ]
     [-lenient]
```

Creates a string that indicates the path of a particular timing exception. You must use the `specify_paths` command as an argument to the `-paths` option in any of the timing exception commands. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these three options.

The `specify_paths` command gives you more detailed control when specifying timing exceptions than the `-from`, `-through`, `-to` options in these timing exceptions could provide.

Options and Arguments

`-capture_clock_pins` *pin*

Specifies a Tcl list of sequential clock pins that capture data. This option can be useful with complex sequential cells such as RAMs that have multiple clock pins.

`-capture_clock_pins_fall_clock` *pin*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies a Tcl list of sequential clock pins that capture data at the falling edge of the ideal clock waveform. This option can be useful with complex sequential cells such as RAMs that have multiple clock pins.

`-capture_clock_pins_fall_pin` *pin*

Specifies a Tcl list of sequential clock pins that capture data at the fall transitions of the specified sequential clock pin. This option can be useful with complex sequential cells such as RAMs that have multiple clock pins.

`-capture_clock_pins_rise_clock` *pin*

Specifies a Tcl list of sequential clock pins that capture data at the rising edge of the ideal clock waveform.

`-capture_clock_pins_rise_pin` *pin*

Specifies a Tcl list of sequential clock pins that capture data at the rise transitions of the specified sequential clock pin.

`-domain` *clock_domain*

Specifies a clock domain the paths should be restricted to.

`-from {`*pin* | *port* | *clock* | *external_delay* | *instance*`}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

Also, if you specify both the `-from` option on an enable pin of a latch and the `-lenient` option, the paths starting on the D pin will also be included in the timing exception.

`-from_fall_clock {`*pin* | *port* | *clock* | *external_delay* | *instance*`}`

Specifies a Tcl list of start points for the paths. The paths are restricted to launch at the falling edge of an ideal clock waveform. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

`-from_fall_pin {`*pin* | *port* | *clock* | *external_delay* | *instance*`}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies a Tcl list of start points for the paths. The paths are restricted to fall transitions at the start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

`-from_rise_clock {`*`pin`* | *`port`* | *`clock`* | *`external_delay`* | *`instance`*`}`

Specifies a Tcl list of start points for the paths. The paths are restricted to launch at the rising edge of an ideal clock waveform. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

`-from_rise_pin {`*`pin`* | *`port`* | *`clock`* | *`external_delay`* | *`instance`*`}`

Specifies a Tcl list of start points for the paths. The paths are restricted to rise transitions at the start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

`-lenient`

Also, if you specify both the `-from` option on an enable pin of a latch and the `-lenient` option, the paths starting on the D pin will also be included in the timing exception.

If a `-from` option is provided that is not a valid start point or a `-to` option is provided that is not a valid end point, then a warning is issued but the rest of the command succeeds. These invalid points are stored in the exception, but will not affect timing analysis results. A warning is also issued when using the `report timing -lint` command in this situation.

`-mode` *`mode_name`*

Indicates the path of a particular timing exception for a specified mode.

`-through {`*`pin`* | *`port`* | *`instance`*`}`

Specifies a Tcl list of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-through_fall_pin {pin | port | instance}`

Specifies a Tcl list of points that a path must traverse. The path is restricted to fall transitions at the indicated points. Points to traverse can be ports, hierarchical pins, pins on a sequential/ mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-through_rise_pin {pin | port | instance}`

Specifies a Tcl list of points that a path must traverse. The path is restricted to rise transitions at the indicated points. Points to traverse can be ports, hierarchical pins, pins on a sequential/ mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-to_fall_clock {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to capture at the falling edge of an ideal clock waveform. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-to_fall_pin {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to fall transitions at the end points. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-to_rise_clock {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to capture at the rising edge of an ideal clock waveform. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-to_rise_pin {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to rise transitions at the end points. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-view view`

Restricts the paths to the specified analysis view.

Examples

- The following example specifies the paths for the `path_disable` timing exception, by specifying a clock pin as start point without any other restrictions:

```
legacy_genus:/> path_disable -paths [specify_paths -from clk1]
```

- The following example creates a group containing paths that are launched by clock `clk1` and which start with a rise pin transition at their start point:

```
legacy_genus:/> path_group -paths [specify_paths -from_rise_pin clk1] -group
I20
```

- The following example uses the `-to_fall_clock` option with a pin object:

```
legacy_genus:/> path_disable -paths [specify_paths -to_fall_clock ff1/D]
```

  The above example specifies that the `path_disable` command should be applied to paths that end at pin `ff1/D` and are captured by a falling edge of an ideal clock waveform.

- Consider an input pin of a RAM that has setup arcs from both pin `CK1` and `CK2`. The following example applies a timing exception to paths using the setup arcs from `CK1`, but not to paths using the setup arcs from `CK2`:

```
legacy_genus:/> path_disable -paths [specify_paths -capture_clock_pins RAM/
CK1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	multi_cycle on page 471
	path_adjust on page 475
	path_delay on page 479
	path_disable on page 482
	path_group on page 485
	report qor on page 690
	report timing on page 709
	write_encounter on page 393
	write_script on page 419
	write_sdc on page 422
Related commands:	create_mode on page 453
	define_clock on page 456
	external_delay on page 464
Affects this attribute:	paths

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

validate_constraints

```
validate_constraints [-rtl] [-netlist string]
     [-init_sequence_file string] [-sdc string]
     [-logfile file] [> file]
```

Validates the SDCs specified in the SDC files against the RTL or netlist. The command can be used any time after elaborating the design. If you do not specify either the `-rtl`, `-netlist`, or `-sdc` options, Genus will validate the internally generated constraints against the design at its current state.

Use this command in the False Path and Multi-cycle Path validation flows.

Options and Arguments

file	Specifies the name of the file to write the report.
`-init_sequence_file` *string*	Specifies the UNIX path to the initialization sequence file for MCP validation.
`-logfile` *string*	Creates a separate CCD logfile. You must specify the UNIX path to the file.
`-netlist` *string*	Specifies the UNIX path to the netlist. This option also indicates that the validation should happen against the netlist (as opposed to the RTL).
`-rtl`	Indicates that the validation should happen against the RTL (as opposed to the netlist).
`-sdc` *string*	Specifies the UNIX path to the SDC files.

Examples

- The following command validates the SDCs in the `top.sdc` file against the RTL:

  ```
  legacy_genus:/> validate_constraints -rtl -sdc /home/test/top.sdc
  ```

- The following command validates the `generic.nl.v` netlist against the `lane.sdc`:

  ```
  legacy_genus:/> validate_constraints -netlist /home/cody/design/netlist.nl.v \
      -sdc /home/cody/lane.sdc
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: generate_constraints on page 468

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

9

Elaboration and Synthesis

- delete_buffer_tree on page 504
- elaborate on page 505
- get_remove_assign_options on page 509
- merge_to_multibit_cells on page 510
- remap_to_dedicated_clock_library on page 512
- remove_assigns_without_optimization on page 514
- remove_clock_reconvergence on page 517
- remove_inserted_sync_enable_logic on page 518
- retime on page 519
- set_remove_assign_options on page 522
- syn_generic on page 526
- syn_map on page 529
- syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_buffer_tree

```
delete_buffer_tree
     [ -drivers {port|pin}...
     | -exclude {port|pin}...]
     [-cone] [-verbose] [design]
```

Removes the buffers and inverter pairs exhaustively or partially (starting from the specified drivers).

This command can for example be used to remove any buffering done on clock paths, or on the paths of DFT control signals.

Options and Arguments

`-cone`	Removes the buffers and inverter pairs starting from the specified drivers to the timing end points. Tshis option is effective only with the -`drivers` option. If you omit this option, deletion of buffers and inverter pairs stops at the first logic instances or sequential instances from the specified drivers.
`design`	Specifies the name of the design from which buffers and inverter pairs need to be removed.
`-drivers`	Specifies the pins and ports from where the buffer tree needs to be removed. By default, all drivers are considered.
`-exclude`	Specifies the drivers from which the buffer tree should not be removed. The buffer tree will be removed from all other drivers.
`-verbose`	Prints information about the buffer trees that are removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

elaborate

```
elaborate [-parameters string] [module]...
     [-libpath path]... [-libext extension]...
```

Creates a design hierarchy consisting of a top-level design and its referenced subdesigns from Verilog or SystemVerilog modules or VHDL entity/architectures. If the top-level design is not specified, it elaborates all the modules not instantiated by any other module as top-level and elaborates them all along with their subdesigns.

The command transforms each module/architecture to be represented as a design in the form of a structural netlist. It also performs semantic checking, sequential register (flops/latches) inferencing, and high-level HDL optimizations. Instances of undefined modules or entities are marked as unresolved and the corresponding modules/entities reported as blackboxes. You can set the attribute `hdl_error_on_latch` to `true`, to error out for latches inferred during elaborate and set the attribute `hdl_error_on_blackbox` to `true`, to error out for blackboxes during elaborate.

The command returns the directory path to the top-level design(s) that it creates.

Note: Before elaborating a design, load your library using the `library` attribute and load your design using the `read_hdl` command into the rc shell.

Options and Arguments

`-libext` *`extension`*	Specifies the extension of the Verilog library files. **Note:** User-specified extensions overwrite the default extensions. *Default*: `.v` and `.V`
`-libpath` *`path`*	Specifies the search path to be used to look for unresolved Verilog module instances. You can specify a relative path with respect to the working directory (.) or an absolute path. **Note:** ~ is currently not supported.
`module`	Specifies the name of either the top-level module or the VHDL configuration to elaborate. If you specify the configuration name of a module instead of the module name, elaboration will occur according to the specified configuration.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you omit *module*, all top-level modules or configurations are elaborated.

Note: When reading in a structural file using the read_hdl `-netlist` command, use this option to specify the top module name.

`-parameters` *string*

Changes the values of Verilog parameters or VHDL generics that are used within the top level code to the specified values.

VHDL generics can be specified in the following forms:
integer: 3, -10
Verilog bit string: 1'b1, 8'hff, 9’so777
boolean: true, false, 1'b1, 1'b0, 1, 0
string literal: {"hello, world”}

Specify the new values in the same sequence as the parameter definitions appear in the code to ensure proper substitution during elaboration.

Genus supports pure and sized integer specifications, and specifications of parameters for submodules and interfaces.

The following example shows a sized integer specification. In this case, 6 is the size (number of bits), d the format (decimal), and 43 the value.

```
elaborate -parameters {6’d43}
```

You can specify a parameter positionally, as an integer in the parameter list, or by name, as a two-element Tcl list. The first element is the name and the second element is the value. For example, you can do either:

```
elaborate -parameters {5 10}
```

```
elaborate -parameters {{width 5} {depth 10}}
```

The following example shows the specification of parameters for submodules or interface instances:

```
elaborate -parameters {{b2.m 0} {b1.w 7’d111}}
```

Examples

- In the following example, the top-level code contains the following parameters in this order:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
parameter data_width1 =  8  ;
parameter data_width2 = 12  ;
parameter averg_period = 4 ;
```

The following command changes `data_width1` to 14, `data_width2` to 12, and `averg_period` to 8 during elaboration:

```
elaborate -parameters {14 12 8} TOP
```

- The following example reads in file `top.v` which has an instance of module `sub`, but file `top.v` does not contain a description of module `sub`.

```
set_attr init_hdl_search_path { ../src ../incl }
elaborate -libpath ../mylibs -libpath /home/verilog/libs -libext ".h"
-libext ".v" top.v
```

The latter command is equivalent to

```
elaborate -libpath { ../mylibs /home/verilog/libs } -libext { ".h" ".v" } top.v
```

First, `elaborate` looks for the `top.v` file in the directories specified through the `init_hdl_search_path` attribute. After `top.v` is parsed, `elaborate` looks for undefined modules (such as `sub`) in the directories specified through the `-libpath` option. First, the tool looks for a file that corresponds to the name of the module appended by the first specified file extension (`sub.h`). Next, it looks for a file that corresponds to the name of the module appended by the next specified file extension (sub.v), and so on.

- Consider the following VHDL example:

```
entity abc is
    generic (str : string := "hello";
             x : bit_vector := "1111";
             w : integer := 5;
             bool : boolean := true);
…
end abc;
```

The following command changes `x` to `"1010"`, `str` to `"hello world"`, `w` to `5` and `bool` to `false`:

```
elaborate -parameters { {x "1010"} {str {"hello world"}} {w 5}  {bool false} } abc
```

- The following command specifies the name of the configuration to elaborate.

```
elaborate $config_name
```

For details, refer Configurations in *Genus HDL Modeling Guide for Legacy UI*.

Related Information

Affected by this command: read_hdl on page 336

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Affected by this attribute: library

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_remove_assign_options

```
get_remove_assign_options
     [-all | -design {design|subdesign]
```

Allows you to check which options you set for the replacement of assign statements on a design or subdesign.

If you did not set any options on a subdesign, this command will not return anything for this subdesign. However if you did set options for the design, these options will be used for all subdesigns unless you specified the `-skip_hierarchical_subdesigns` option.

Options and Arguments

`-all`	Returns settings for the design and all subdesigns.
`-design {design | subdesign}`	Returns the settings for the design or specified subdesign.

Related Information

Affected by these commands: remove_assigns_without_optimization on page 514
set_remove_assign_options on page 522

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

merge_to_multibit_cells

```
merge_to_multibit_cells
     [[-sequential] [-combinational] [-iso_ls] | -all]
     [-logical] [-single_thread] [design]
```

Performs standalone mapping of single bit cells to multibit cells without performing any other optimizations.

The recommended multibit merge flow is to do multibit merge in `syn_map` stage. Set the `use_multibit_cells` attribute to `true` and other multibit attributes before `syn_map`, then `syn_map` will do multibit merge at the last stage.

Genus multibit merge is timing driven.

You can use the `merge_to_multiple_cells` command to perform standalone multibit merge.

You can set different multbit attributes before `merge_to_multiple_cells` command to control the multibit merging. For example, you can define `dont_merge_multibit` attribute on the single bit cells for which merging in not required.

After completion of `merge_to_multiple_cells` command, `report multibit_inferencing -not_merged_instance_list` command can be used to report all the not merged single bit cells and their corresponding reason for not getting merged by `merge_to_multiple_cells` command. Not merged single bit cells might have multiple reasons for not getting merged but, `report multibit_inferencing -not_merged_instance_list` command will report only one reason per not merged single bit cell.

`multibit_rejection_reason` attribute can be checked on not merged single bit cells to report the reason for not getting merged by `merge_to_multiple_cells` command.

Note: This command can only be run on a mapped design.

Options and Arguments

`-all`	Allows multibit mapping of combinational and sequential cells, and isolation cells and level-shifter cells.
`design`	Specifies the name of the design for which to perform multibit mapping.
`-combinational`	Allows multibit mapping of combinational cells.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-iso_ls`	Allows multibit mapping of isolation cells and level-shifter cells.
`-logical`	Allows logical-only merging, thereby ignoring placement data even if it is available. By default, the tool performs placement-based multibit merging if the design is placed.
`-sequential`	Allows multibit mapping of sequential cells.
`-single_thread`	Performs single-threaded multibit mapping. By default, the tool performs super-threaded multibit mapping if the number of sequential instances is larger than 256.

Examples

```
set_attr use_multibit_cells true
merge_to_multibit_cells -logical $DESIGN
```

Related Information

Affects this command:	report multibit_inferencing on page 650
Affects this attribute:	multibit_rejection_reason
Affected by these attributes:	use_multibit_cells
	use_multibit_combo_cells
	use_multibit_seq_and_tristate_cells
	physical_aware_multibit_mapping
	force_merge_seqs_into_multibit_cells
	multibit_cells_from_different_busses
	multibit_auto_exclude_registers_with_exceptions
	multibit_allow_unused_bits
	dont_infer_multibit
	multibit_allow_async_phase_map
	multibit_mapping_effort_level

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remap_to_dedicated_clock_library

```
remap_to_dedicated_clock_library
     [-clocks clock...] [-dont_add_preserve]
     [-effort {low|medium|high}] [design]
```

Remaps the instances in the clock path of the specified clock(s) to the library cells specified through the `clock_library_cells` clock attribute.

Note: In Genus, clock networks are considered ideal. If preserve is set to true on the clock path elements, use `set_attribute ui_respects_preserve false` before running the command.

Options and Arguments

`-clocks clock`
: Specifies the name of the clocks for which to map the clock-path logic to its library cell collection.

design
: Specifies the design for which to perform the remapping.

`-dont_add_preserve`
: Prevents that the instances on the clock path are marked preserved.

  By default the `preserve` attribute is set to `true` for the instances on the clock path.

`-effort {low|medium|high}`
: Specifies the effort level for remapping and resizing.

  - `low`—Performs a one-to-one replacement.

    This works well when the given set of cells is sufficient to replace the complete clock-path logic.
  - `medium`—Performs partial one-to-one replacement.

    Replaces only those instances which have a suitable replacement in the provided library set for the clock.
  - `high`—Performs partial one-to-one replacement and remapping. Replaces only those instances which have suitable replacement in the provided clock library set, and resynthesizes or remaps the remaining instances using available libcells in the library set (remapping).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: `medium`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_assigns_without_optimization

```
remove_assigns_without_optimization
     [-buffer_or_inverter <libcell>]
     [-no_buffers_use_inverters]
     [-allow_unloaded_buffers]
     [-use_inverted_signal]
     [-ignore_preserve_map_size_ok]
     [-ignore_preserve_setting]
     [-include_local_constant_assigns]
     [-dont_respect_boundary_optimization]
     [-dont_skip_unconstrained_paths]
     [-preserve_dangling_nets]
     [-skip_hierarchical_subdesigns]
     [-ports ports] [-verbose]
     [-design {subdesign|design}]
```

Controls the aspects of the replacement of `assign` statements in the design with buffers or inverters without performing incremental optimization.

To avoid incremental optimization when removing assign statements, make sure that the `remove_assigns` attribute is set to `false`, and specify this command after synthesis.

Important

Make sure to uniquify the design before you run this command, otherwise the tool will issue an error message.

In the Physical flow, Genus will by default not remove the assign statements inserted by Innovus on constants, because inserting extra buffers to remove assigns on constants could lead to higher congestion.

To remove assigns on constants after the Physical flow, use the following command:

```
remove_assigns_without_optimization -dont_skip_unconstrained_paths -design design
```

Note: The command will not remove assignment statements when this could lead to an NEQ.

Options and Arguments

`-allow_unloaded_buffers`

Prevents unloaded nets from being deleted and allows unloaded buffers to be added on these nets to fix the assigns in the netlist.

`-buffer_or_inverter` *`libcell`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the buffer or inverter to be used to replace the `assign` statements. The specified cell must be part of a library that was loaded.

If this option is omitted, Genus will determine which buffers or inverters to insert. If a particular library domain does not have any buffers or inverters, no buffers or inverters will be added in that domain. However, buffers or inverters will be added in other domains depending on availability.

`-design {`*`design`* `|` *`subdesign`*`}`

Indicates that the specified command options apply to the specified design or subdesign. If neither is specified, the options apply to the entire design.

`-dont_respect_boundary_optimization`

Allows Genus to disconnect the unloaded subports of a subdesign, if the `boundary_opto` attribute on the subdesign was set to `false`.

`-dont_skip_unconstrained_paths`

Specifies to remove assigns on unconstrained paths. These paths can be paths driven by a constant, false paths with assigns, or paths without any timing constraints.

By default, Genus skips unconstrained paths.

`-ignore_preserve_map_size_ok`

Allows assign statements in the module to be removed if the `preserve` attribute on the module is set to `map_size_ok`.

`-ignore_preserve_setting`

Allows assign statements in the module to be removed independent of the setting of the `preserve` attribute on the module.

`-include_local_constant_assigns`

Allows to remove local constant driven assigns that have multiple drivers and do not drive hierarchical ports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-no_buffers_use_inverters`

Specifies to search for inverters and to use them in case the library or library domain does not have any buffers. If buffers exist they will be used, irrespective of whether the library has inverters or not.

`-ports` *`ports`*

Limits the replacement of `assign` statements to the specified ports.

`-preserve_dangling_nets`

Specifies to preserve a net that is driven by a buffer or inverter that is inserted when assign statements are replaced and if that net does not have a fanout in the next level of hierarchy and was not driven by a constant before adding the buffer or inverter.

`-skip_hierarchical_subdesigns`

Restricts the replacement of `assign` statements to one level only. By default, the replacement of `assign` statements is done recursively.

`-use_inverted_signal`

Allows the tool to use an inverter with inverted signal if the following conditions are met:

- The assign statement is driven by a constant
- The area of the inverter is smaller than the area of the buffer

`-verbose`

Displays all messages during assign removal.

Examples

- The following example removes assigns on unconstrained ports:

```
remove_assigns_without_optimization -dont_skip_unconstrained_paths
```

- .The following example removes assigns at port out1:

```
remove_assigns_without_optimization -port out1
```

Related Information

Affected by this attribute: remove_assigns

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_clock_reconvergence

```
remove_clock_reconvergence
     [-clocks clock...]  [-effort {low|high}]
     [-ports port_list] [design]...
```

Optimizes the clock path by removing clock path re-convergence for all clocks. You can also remove the reconvergence for the specified clock only.

Options and Arguments

`-clocks` *`clock`*	Specifies the name of the clocks for which to remove the clock path reconvergence. By default, the command optimizes the networks for all clocks.
`design`	Specifies the design for which to perform the removal of clock reconvergence.
`-effort {low|high}`	Specifies the effort level during optimization. `low`—Performs clock reconvergence removal. `high`—Performs clock reconvergence removal and in addition fixes common clock inputs and constant inputs. *Default:* `low`
`-ports` *`port_list`*	Removes the reconvergence through the specified ports.

Examples

- The following command removes the reconvergence for clock `clk1` and, in addition, fixes common clock inputs and constant inputs.

  ```
  remove_clock_reconvergence -clocks [find / -clock clk1] -effort high
  ```

- The following command removes the reconvergence through the `pcm_sclk` and `pcm_lrclk` ports.

  ```
  remove_clock_reconvergence -ports "[find / -port pcm_sclk] \
  [find / -port pcm_lrclk]"
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_inserted_sync_enable_logic

`remove_inserted_sync_enable_logic [> file]`

Removes timing critical synchronous enable logic from flops.

Tools like PowerPro™ from Calypto™ can insert synchronous enable logic for sequential clock-gating in the RTL code and write out optimized RTL. However if the synchronous enable logic is inserted on a timing critical path, Genus can remove the synchronous enable logic to improve overall timing of the design.

Use the `remove_inserted_sync_enable_logic` command after mapping and before incremental or after incremental optimization.

By disconnecting the synchronous enable, some sequential instances can become unloaded. Run incremental optimization (`synthesize -incremental`) to remove these unloaded sequential instances.

Options and Arguments

file	Specifies the name of the file to which Genus must write the list of flops from which the synchronous enable pins were removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

retime

```
retime [-prepare] [-min_area] [-min_delay]
     [-effort {medium | low | high}]
     [-clock clocks]
     [design | subdesign]...
```

Improves the performance of the design by either optimizing the area or the clock period (timing) of the design.

The recommended retiming flow is automatic retiming flow which is implemented by setting the retime attribute to true on the dedicated design before syn_generic command. Retiming is performed automatically on the blocks marked with retime attribute during generic synthesis.

The retime command is used for debugging purpose, for example, if a specific block in the design is not retimed, the command can be used to debug. Perform block-level retiming on a block-by-block basis to further optimize the design, thereby minimizing the delay or area. This command should be executed on a post elaboration design. If no option is specified for the command, the -`min_delay` option is implied.

Optimization is realized through appropriately moving registers.The area optimization will not degrade the timing.

Generally, you should run `retime -prepare` first as it prepares the design for retiming by deriving the appropriate constraints and synthesizing to a gate-level design that is ready for retiming. If there are some potential problems in the design, `retime -prepare` will give some Errors or Warnings. After `retime -prepare` is successfully executed, you can then run other retime commands such as `retime -min_delay/min_area`.

Options and Arguments

design|*subdesign*	Specifies the name of the design or subdesign you want to retime.
`-clock` *clocks*	Specifies to perform retiming on the registers clocked by the specified clocks
`-effort {medium | low | high}`	The effort levels are only available for the `min_delay` option. ■ `high` — Genus consumes as much time as necessary to provide the optimum timing solution.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `low` — Provides a rough retiming estimate. This effort level provides the quickest results among the three effort levels.
- `medium` — Provides results that are approximately within 1% of the optimum solution.

Default: `medium`

`-min_area` Optimizes the design for area by minimizing the number of registers without degrading the critical path in the design.

`-min_delay` Optimizes the design for timing. For the best results, you should first issue the `retime -prepare` command separately before issuing the `retime -min_delay` command.

`-prepare` Prepares the design for retiming and then synthesizes the design. Specifically, the `retime -prepare` command prepares the design by constraining paths according to the path delays through registers (from inputs to outputs) as opposed to register to register. There is no need to separately issue the `synthesize` command after issuing the `retime -prepare` command.

Examples

- The following example retimes the design to optimize for timing:

```
...
legacy_genus:/> retime -prepare
legacy_genus:/> retime -min_delay
...
```

Related Information

Related command:	syn_generic on page 526
Affected by these attributes:	dont_retime
	retime
	retime_async_reset
	retime_effort_level
	retime_hard_region
	retime_move_mux_loop_with_reg

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related attributes:

retime_optimize_reset

retiming_clocks

retime_original_registers

retime_reg_naming_suffix

retime_verification_flow

trace_retime

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_remove_assign_options

```
set_remove_assign_options [-design {design | subdesign}]
     [ [-buffer_or_inverter libcell]
       [-allow_unloaded_buffers]
       [-no_buffers_use_inverters]
       [-use_inverted_signal]
       [-ignore_preserve_setting]
       [-ignore_preserve_map_size_ok]
       [-include_local_constant_assigns]
       [-preserve_dangling_nets]
       [-dont_respect_boundary_optimization]
       [-dont_skip_unconstrained_paths] [-verbose]
       [-skip_hierarchical_subdesigns]
     | -reset]
```

Controls the aspects of the replacement of `assign` statements in the design with buffers or inverters, which is controlled by the `remove_assigns` root attribute. The actual replacement happens during the next incremental optimization run.

The replacement of `assign` statements is performed

- Only on the objects (design or subdesigns) specified with the `-design` option
- Only once during each incremental optimization run operation

  If your script contains multiple occurrences of the `set_remove_assign_options` command with different settings for the same object, the last settings before the next synthesize command will prevail.
- On the objects in the order of their corresponding `set_remove_assign_options` commands.

The specified options apply to all subsequent incremental optimization runs unless you reset the options with the `-reset` option.

Options and Arguments

`-allow_unloaded_buffers`

Prevents unloaded nets from being deleted and allows unloaded buffers to be added on these nets to fix the assigns in the netlist.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-buffer_or_inverter` *`libcell`*

Specifies the buffer or inverter to be used to replace the `assign` statements. The specified cell must be part of a library that was loaded.

If this option is omitted, Genus will determine which buffers or inverters to insert. If a particular library domain does not have any buffers or inverters, no buffers or inverters will be added in that domain. However, buffers or inverters will be added in other domains depending on availability.

`-design {`*`design`* `|` *`subdesign`*`}`

Indicates that the specified command options apply to the specified design or subdesign. If neither is specified, the options apply to the entire design.

`-dont_respect_boundary_optimization`

Allows Genus to disconnect the unloaded subports of a subdesign, if the `boundary_opto` attribute on the subdesign was set to `false`.

`-dont_skip_unconstrained_paths`

Specifies to remove assigns on unconstrained paths. These paths can be paths driven by a constant, false paths with assigns, or paths without any timing constraints.

By default, Genus skips unconstrained paths.

Note: This option has no effect in the Genus-Physical flow.

For place and route tools removing assign statements is only necessary on constrained paths because these paths might not be optimized well with assigns present.

Removing assign statements on unconstrained paths like constant driven nets/hierarchical ports might cause placement challenges during legalization in the Genus-Physical flow as these nets and ports don't have any anchors to guide the placement engine.

Consequently the tool does not remove assign statements on unconstrained paths. This should not have any effect on the backed tools. Genus handles assign statements natively and does not require assign removal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: To remove unconstrained paths in the Genus-Physical flow use `remove_assigns_without_optimization -dont_skip_unconstrained_paths` after `synthesize -to_placed`.

`-ignore_preserve_map_size_ok`

Allows `assign` statements in the module to be removed if the `preserve` attribute on the module is set to `map_size_ok`.

`-ignore_preserve_setting`

Allows `assign` statements in the module to be removed independent of the setting of the `preserve` attribute on the module.

`-include_local_constant_assigns`

Allows to remove local constant driven assigns that have multiple drivers and do not drive hierarchical ports.

`-no_buffers_use_inverters`

Specifies to search for inverters and to use them in case the library or library domain does not have any buffers. If buffers exist they will be used, irrespective of whether the library has inverters or not.

`-preserve_dangling_nets`

Specifies to preserve a net that is driven by a buffer or inverter that is inserted when `assign` statements are replaced and if that net does not have a fanout in the next level of hierarchy and was not driven by a constant before adding the buffer or inverter.

`-reset` Specifies to use the command with the default settings.

`-skip_hierarchical_subdesigns`

Restricts the replacement of `assign` statements to one level only. By default, the replacement of `assign` statements is done recursively.

`-use_inverted_signal`

Allows to use a single inverter with an inverted signal instead of a buffer if the inverter cell is smaller in area.

`-verbose` Displays all messages during `assign` removal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command specifies to use buffer `BUFX2` to replace `assign` statements in subdesign `med`.

```
legacy_genus:/> set_remove_assign_options -buffer BUFX2 [find / -subdesign
med]
```

- In the following example, the subdesigns `bottom` and `top` represent two different library domains. Two different kinds of buffers are specified for each domain.

```
legacy_genus:/> set_remove_assign_options -buffer BUFX2 [find / -subdesign
bottom]
legacy_genus:/> set_remove_assign_options -buffer BUFX7 [find / -subdesign
top]
```

- In the following example, several settings are specified for subdesign `sub` before the synthesize command.

```
set_remove_assign_options -buffer_or_inverter [ find / -libcell buf1] \
-design {find / -subdesign sub]
set_remove_assign_options -buffer_or_inverter [ find / -libcell inv] \
-design sub
synthesize -incr
```

  Only the second `set_remove_assign_options` command is taken into account during the next optimization run and no `assign` removals are performed at the top level.

- In the following example, several incremental optimization runs are performed.

```
set_remove_assign_options -design top
synthesize -incr
set_remove_assign_options -reset -design top -dont_skip_unconstrained_paths
synthesize -incr
```

  During the first optimization run, unconstrained paths remain untouched in design `top`. However, before the second optimization run starts the options have been reset for design `top` and `assign` statement can now be removed from the unconstrained paths.

Related Information

Affects this command:	syn_opt on page 531
Affected by this attribute:	remove_assigns

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

syn_generic

```
syn_generic
     [-physical] [design]...
```

Takes an elaborated and fully constrained design as input and synthesizes it into a netlist of generic gates by doing high-level RTL and datapath optimizations.

Main steps include:

- Optimization of datapath components (adders, multipliers, shifters, etc.) to implement arithmetic modules that meet the required performance goal while minimizing area and power. You can control the global trade-off between timing and area by using the `dp_analytical_opt` attribute. You can have more control over specific datapath transformations through datapath attributes.
- Implementation of mux structures and choosing the best architecture (binary or one-hot) for them.
- Using enable conditions in the fanin of registers to implement clock-gating logic and thus reducing power. The main attribute to enable this feature is `lp_insert_clock_gating` (default is off). For details refer to the other related attributes in the `lp_cg` category.
- Apply retiming to improve overall maximum performance of the design. This technique is applied to all modules tagged with the retime attribute. Further control is possible via attributes of the retime category.

It is recommended to use the `check_design` command in advance to find major design problems like unresolved modules and multi-driven or undriven nets or pins. Also, the command `check_timing_intent` helps to identify issues related to the timing constraints read into the tool.

If the -`physical` option is specified, Genus will also place the generic netlist and compute an initial congestion map based on actual global routing that enables the user to rate the quality of the given floorplan. Top paths are physically optimized, so timing bottlenecks of the design can be identified quickly. The resulting database can be used to analyze congestion and critical path placement even before mapping to the technology library. Use the `check_floorplan` command in advance to ensure the given floorplan data is of high quality.

The effort of this command is controlled via the `syn_generic_effort` attribute.
Use the `write_snapshot` command to capture relevant metrics after running syn_generic.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Large parts of generic optimization are either multi-threaded or executed via partition-based synthesis, distributed across multiple machines and/or CPUs. By increasing the number of CPUs the user can enable Genus to better balance the workload and improve runtime further.

Options and Arguments

design	Specifies the name of the design that will be synthesized. If you omit the design name, the top-level design of the current directory of the design hierarchy is used. If multiple top-level designs exist, all are synthesized.
`-physical`	Instructs Genus to do physical-aware generic optimization, including global placement and routing of the design. If not specified, Genus will do logical optimization only. If the design attribute `physical_aware_mapping` is set to true, the tool does accurate long wire prediction that influences datapath structuring and cell selection for convergent timing closure. **Note:** This option is available only with the Genus_Physical_Opt License Key.

Example

- The following example runs physical-aware generic optimization on the top-level design in memory:

```
syn_generic -physical
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Performing Synthesis in *Genus User Guide for Legacy UI.*

Genus Datapath Synthesis Guide for Legacy UI

Retiming the Design chapter in the *Genus User Guide for Legacy UI*

Affects these commands	report area on page 584
	report clock_gating on page 593
	report datapath on page 603
	report design_rules on page 608
	report gates on page 633
	report power on page 669
	report summary on page 703
	report timing on page 709
Related commands	syn_map on page 529
	syn_opt on page 531
Affected by this attribute:	syn_generic_effort

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

syn_map

```
syn_map
     [-physical] [design]...
```

Maps a design from generic gates to a technology library while optimizing for best performance, power and area. Genus evaluates multiple implementations for a given logic cone and chooses the one that meets timing constraints while minimizing area and power. After an initial mapping phase, further refinement is done by incrementally optimizing the netlist to recover area and power while maintaining timing.

Main steps include:

- Structuring and mapping of generic logic to meet specified goals. Certain features of optimization can be controlled via attributes.
- Selective ungrouping of modules to improve optimization for best timing or area. This can be controlled via the `auto_ungroup` attribute.
- Optimizing for leakage and dynamic power. The effort spent on reducing leakage power is specified via the `leakage_power_effort` attribute. By default, Genus does not optimize for power.
- Mapping of single-bit to multi-bit library cells. By default this feature is turned off, but it can be enabled via the `use_multibit_cells` attribute.

The library cell set available to the mapper can be restricted by applying the `dont_use` attribute to library cell objects. It is recommended to only apply this attribute in case certain library elements are not allowed to be used under any circumstances and not as an overall guidance to the mapping engine.

If the `-physical` option is specified, Genus will perform physical-aware mapping and restructuring of logic to account for physical effects like long wires in the design. Placement locations of registers are maintained throughout mapping to allow for physical-aware multi-bit merging of flip flops and physical-aware scan insertion post `syn_map`. It is a prerequisite to run physical-aware mapping if `syn_generic` was run in physical mode too.

The effort to use during optimization is controlled by the `syn_map_effort` root attribute. Use the `write_snapshot` command to capture relevant metrics after running `syn_map`.

Large parts of mapping are either multi-threaded or executed via partition-based synthesis, distributed across multiple machines and/or CPUs. By increasing the number of CPUs, you can enable Genus to better balance the workload and improve runtime further.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

design	Specifies one or multiple designs to be synthesized. If you omit the design name, the top-level design of the current directory of the design hierarchy is used. If multiple top-level designs exist, all are synthesized.
`-physical`	Instructs Genus to perform physical-aware mapping. If not specified, Genus will only do logical optimization. **Note:** This option can be used if and only if `syn_generic -physical` was done previously **Note:** This option is only available with the `Genus_Physical_Opt` License Key.

Example

- The following example runs physical-aware mapping on the top-level design in memory:

```
syn_map -physical
```

Related Information

Performing Synthesis in *Genus User Guide for Legacy UI.*

Affects these commands	report area on page 584
	report datapath on page 603
	report design_rules on page 608
	report gates on page 633
	report power on page 669
	report summary on page 703
	report timing on page 709
Related commands	syn_generic on page 526
	syn_opt on page 531
Affected by this attribute:	syn_map_effort

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

syn_opt

```
syn_opt [-physical] [-spatial]
     [-incremental | -2d_compression] [design]...
```

Takes a mapped design as input and incrementally optimizes timing, area and power. Without specifying the spatial or physical flow options, syn_opt will do pure logical optimization, even if called on a design database that was generated by physical-aware mapping. Depending on the effort level specified by the syn_opt_effort attribute, it first optimizes worst and total negative slack, then applies certain transforms to recover area and power. The higher the effort, the more iterations of optimization steps are done.

Both physical-aware flows distinguish themselves by:

- The spatial flow initially runs the Genus native placer, followed by physical-aware incremental optimization in Genus. The outcome is a placed and optimized design, which can be used to measure pre-CTS performance and routability. The resulting design is ready to be passed onwards to Innovus for a full pre-CTS optimization step.
- The physical flow passes the design to Innovus for placement, DRV fixing and an initial round of physical optimization. After bringing back the design into Genus, further passes of incremental optimization are done in Genus. The resulting design is ready to be passed onwards to Innovus for an incremental pre-CTS optimization step.

Multiple features of incremental optimization (e.g. timing-driven duplication of registers) can be controlled via attributes, for a full list of these check the iopt category.

The effort of this command is controlled via he `syn_opt_effort` root attribute. Use the write_snapshot command to capture relevant metrics after running syn_opt.

Large parts of incremental optimization are either multi-threaded or executed via partition-based synthesis, distributed across multiple machines and/or CPUs. By increasing the number of CPUs the user can enable Genus to better balance the workload and improve runtime further.

Options and Arguments

`-2d_compression`	Implements a two-dimensional elastic compression logic for testability, which has been configured in advance via the relevant DFT commands and attributes.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	This option can be used with either `spatial` or `physical` flow options but not using the `incremental` mode. It also is not allowed in combination with `express` effort mode (set through the `syn_opt_effort` attribute).
design	Specifies the name of the design to be synthesized. If omitted, the top-level design of the current directory in the design hierarchy is used. If multiple designs exist, all of them are synthesized.
`-incremental`	This option can be used to do further incremental refinement after an initial `syn_opt` call has been executed. It works with either `logical` or `physical-aware` flows. **Note:** `-physical` option has to be used if `syn_opt -physical` was done previously. That is, `syn_opt -physical -incremental`.
`-physical`	Instructs Genus to run the physical flow, utilizing Innovus placement and optimization in combination with Genus native physical optimization features. Requires a full Innovus installation to be present in your search path. Performs placement of mapped gates and placement-based optimizations. **Note:** This option is only available with the Genus_Physical_Opt License Key. **Note:** You must have access to the Innovus Implementation System to run this command option. **Note:** It is highly recommended to use the same versions of Genus and Innovus.
`-spatial`	Instructs Genus to run the spatial flow to get an accurate prediction of pre-CTS performance of the design.

Examples

- The following example runs a pure logical optimization:

```
syn_opt
```

- The following example run a spatial flow with 2D Elastic compression insertion:

```
syn_opt -spatial -2d_compression
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands	report area on page 584
	report datapath on page 603
	report design_rules on page 608
	report gates on page 633
	report power on page 669
	report summary on page 703
	report timing on page 709
Related commands	remove_assigns_without_optimization on page 514
	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	syn_opt_effort
	iopt_force_constant_removal
	iopt_sequential_resynthesis
	iopt_sequential_resynthesis_min_effort

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

10

Hierarchical Flow

- assemble_design on page 536
- generate_ilm on page 538
- read_ilm on page 540
- read_ilm_from_files on page 541

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

assemble_design

```
assemble_design [design]
     [-config_view_name_mapping string]
```

Integrates the interface logic models in the design.

Use this command after elaborating the design which includes the top level module and all the ILMs. The DEF file for the top-level design is also required before assembling the design if physical ILM are read (using the `read_ilm` command without the `-logical` option).

Note: In case of a multi-mode design, you must create the modes and read the top-level SDC file before assembling the design.

Options and Arguments

`-config_view_name_mapping` *`string`*

Specifies a string for name mapping if view name given in ILM configuration file is different from the view name at TOP level. ILM view name should be mentioned first.
For example: `{`*`ilm_view_1 top_view_1`*`}` `{`*`ilm_view_2 top_view_2`*`}`

`design`

Specifies the name of the design for which to do the checking.

This argument is only required if multiple designs are loaded.

Examples

- Logical flow:

```
read_hdl top.v
read_ilm -basename ilm_dir/leon -module leon -logical
read_ilm -basename ilm_dir/dsp -module dsp -logical
elaborate top
assemble_design
```

- Physical flow:

```
read_hdl top.v
read_ilm -basename ilm_dir/leon -module leon
read_ilm -basename ilm_dir/dsp -module dsp
elaborate top
read_def -design top top.def
assemble_design
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related commands: generate_ilm on page 538
read_ilm on page 540

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

generate_ilm

```
generate_ilm
     {-basename string |-preview}
     [-latch] [-gzip_files] [design]
```

Generates an interface logic model (ILM) for the design. The ILM contains:

- A Verilog structural netlist with information about the instances associated with the interface logic.
- A DEF file containing the physical information of the nets, interconnects, pins, and the other hard macros in the design interface.

  A DEF file is only generated if placement data is available.
- A SPEF file that contains parasitic information of the pins and nets at the design interface
- An SDC file that contains `set_transition` and `set_case_analysis` information for the inputs, nets and instance pins of the interface logic. This allows for accurate calculation of delays and slews.

In Genus, you should generate an ILM for the design at the end of the synthesis run.

Note: When generating an ILM, any logic that is not part of an I/O path will be removed from the database and the design will be reduced to an interface logic model. Therefore, you should save a snapshot of the design (`write_db`) prior to generating the ILM.

Options and Arguments

`-basename` *`string`*	Specifies the directory path name and base filename for the output data. This option is not required if you generate a preview.
`design`	Specifies the name of the design for which to generate the model. This argument is only required if multiple designs are loaded.
`-gzip_files`	Compresses the generated files in gzip format.
`-latch`	Specifies whether the ILM should stop at the first latch. By default, the command will include latches in the interface logic and stop at the first flop.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-preview`	Only generates a reduction summary report which gives you a comparison between the number of instances and registers in the original design and in the to be generated ILM. Based on the reported reduction ratio, you can decide whether to generate the ILM for the block.

Related Information

Related commands:	assemble_design on page 536
	read_ilm on page 540

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_ilm

`read_ilm -basename string -module_name module [-logical]`

Reads in the interface logic model (ILM) for the specified module.

You should read in the ILMs after reading in the RTL of the design, that is, before elaborating the design.

Options and Arguments

`-basename string`	Specifies the directory path name and base filename for the ILM data.
`-logical`	Loads only the logical data.
`-module_name module`	Specifies the name of the ILM.

Related Information

Related commands:	assemble_design on page 536
	generate_ilm on page 538

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_ilm_from_files

```
read_ilm_from_files
     -module_name module -verilog_file file
     -input_directory string [-logical]
     [-output_directory string] [-prefix string]
     [-def_file file] [-spef_file file]
     [-sdc_file file]
     [-sdc_by_mode {{mode_name_1 sdc_file}
       [{mode_name_2 sdc_file}]...}]
```

Imports an interface logic model (ILM) for the specified module.

Note: Provide at least an SDC file or SPEF file for the ILM, but it is recommended to specify both.

Options and Arguments

`-def_file` *file*	Specifies the name of the DEF file corresponding to the ILM.
`-input_directory` *string*	Specifies the directory containing the ILM data to be imported
`-logical`	Loads only the logical data.
`-module_name` *module*	Specifies the name of the ILM.
`-output_directory` *string*	Specifies the directory where Genus should write out the tool-generated ILM data.
`-prefix` *string*	Specifies the prefix for the ILM data generated by Genus. *Default*: the specified module name.
`-sdc_by_mode` *string*	Specifies a list of Tcl lists. Each Tcl l list contains a mode of the ILM with the corresponding SDC file.
`-sdc_file` *file*	Specifies the name of the SDC file corresponding to the ILM.
`-spef_file` *file*	Specifies the name of the SPEF file corresponding to the ILM.
`-verilog_file` *file*	Specifies the name of the Verilog file of the ILM.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

11

Analysis and Report

- all on page 547
- all_connected on page 548
- all des on page 549
- all des inps on page 550
- all des insts on page 551
- all des outs on page 552
- all des seqs on page 553
- all lib on page 555
- all lib bufs on page 556
- all lib ties on page 557
- analyze_library_corners on page 558
- check_design on page 560
- check_timing_intent on page 565
- clock_ports on page 567
- compare_sdc on page 568
- create_timing_bin on page 569
- fanin on page 572
- fanout on page 576
- report on page 579
- report area on page 584
- report boundary_opto on page 587

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- report case_analysis on page 589
- report cdn_loop_breaker on page 590
- report cell_delay_calculation on page 592
- report clock_gating on page 593
- report clocks on page 600
- report congestion on page 602
- report datapath on page 603
- report design_rules on page 608
- report dft_chains on page 610
- report dft_clock_domain_info on page 615
- report dft_core_wrapper on page 617
- report dft_registers on page 620
- report dft_setup on page 624
- report dft_violations on page 629
- report disabled_transparent_latches on page 632
- report gates on page 633
- report hierarchy on page 637
- report instance on page 639
- report memory on page 642
- report memory_cells on page 643
- report messages on page 644
- report min_pulse_width on page 647
- report mode on page 648
- report module on page 649
- report multibit_inferencing on page 650
- report net_cap_calculation on page 657
- report net_delay_calculation on page 658

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- report net_res_calculation on page 660
- report nets on page 661
- report opcg_equivalents on page 664
- report ple on page 666
- report port on page 668
- report power on page 669
- report power_intent on page 681
- report power_intent_instances on page 685
- report proto on page 688
- report qor on page 690
- report scan_compressibility on page 697
- report sequential on page 699
- report slew_calculation on page 702
- report summary on page 703
- report test_power on page 705
- report timing on page 709
- report units on page 721
- report utilization on page 722
- report yield on page 723
- report_chains_longest_wires on page 724
- report_chains_total_wirelength on page 726
- report_chains_wires on page 728
- report_cwd_check on page 730
- report_dft_abstract_models on page 731
- report_dft_trace_back on page 732
- report_flow on page 734
- report_logic_levels_histogram on page 735

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- report_metric on page 736
- report_obj on page 737
- report_scan_compression_logic on page 739
- report_signoff_timing on page 741
- statistics on page 742
- statistics add_metric on page 744
- statistics log on page 746
- statistics read on page 748
- statistics remove_metric on page 749
- statistics report on page 750
- statistics reset on page 753
- statistics run_stage_ids on page 754
- statistics write on page 755
- time_info on page 756
- timestat on page 757
- validate_timing on page 758

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all

```
all {des | lib}
```

Generates a Tcl list of the specified objects. For more information on specific `all` commands, see Related Information.

Options and Arguments

`des`	Generates a list of various design objects.
`lib`	Generates a list of various libcell objects.

Related Information

Related commands:

all des on page 549
all des inps on page 550
all des insts on page 551
all des outs on page 552
all des seqs on page 553
all lib on page 555
all lib bufs on page 556
all lib ties on page 557

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_connected

`all_connected [-leaf_pin] {net | pin | pgpin | port}...`

If the specified object is a net, the command returns the list of all the pins connected to the net. If the object is a pin or a port, the command returns the net connected to the pin or the port.

Options and Arguments

`-leaf_pin`	Returns only the leaf cell pins connected to the specified net.
net	Specifies a net. The ensuing list will return a list of all the pins connected to this net.
pin	Specifies a pin. The ensuing list will return the net connected to the pin.
pgpin	Specifies a power or ground pin. The ensuing list will return the net connected to the pin.
port	Specifies a port. The ensuing list will return the net connected to the port.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all des

```
all des {inps | insts | outs | seqs}
```

Generates a Tcl list based on the specified object. For more information on specific `all des` commands, see Related Information.

Options and Arguments

`inps`	Generates a list of all input ports of the specified clock or clock domain.
`insts`	Generates a list of all the instances in the design.
`outs`	Generates a list of all output ports of the specified clock or clock domain.
`seqs`	Generates a list of all the sequential instances in the design.

Related Information

Related commands:

all des inps on page 550
all des insts on page 551
all des outs on page 552
all des seqs on page 553

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all des inps

```
all des inps
     [-clock clock...] [-clock_domains clock_domain...]
     [design]
```

Generates a Tcl list of all input ports of the specified clock or clock domain.

Options and Arguments

`-clock` *`clock`*	Returns a list of input ports for the specified clock or clocks.
`-clock_domains` *`clock_domain`*	Returns a list of input ports for the specified clock domain or domains.
`design`	Returns a list of input ports for the specified design. If a design is not specified, the input ports for the current design will be returned.

Example

- The following example returns all the input ports for the clock named `clock1`.

```
legacy_genus:/> all des inps -clock clock1
{/designs/PENNY/ports_in/in1[3]} {/designs/PENNY/ports_in/in1[2]}
{/designs/PENNY/ports_in/in1[1]} {/designs/PENNY/ports_in/in1[0]}
{/designs/PENNY/ports_in/in2[3]} {/designs/PENNY/ports_in/in2[2]}
{/designs/PENNY/ports_in/in2[1]} {/designs/PENNY/ports_in/in2[0]}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all des insts

`all des insts [-unresolved] [design]`

Generates a Tcl list of all instances in the specified design. You can return a list of only unresolved instances by specifying the `-unresolved` option.

Options and Arguments

design	Returns a list of instances for the specified design. If a design is not specified, the instances for the current design will be returned.
`-unresolved`	List only unresolved instances and omit everything else.

Example

- The following example returns a list of all instances in the design named `STONE`.

```
legacy_genus:/> all des insts STONE
/designs/STONE/instances_hier/inst1
/designs/STONE/instances_hier/inst1/instances_comb/g1
/designs/STONE/instances_hier/inst1/instances_comb/g2
/designs/STONE/instances_hier/inst1/instances_comb/g3
/designs/STONE/instances_hier/inst1/instances_comb/g4
/designs/STONE/instances_hier/inst2
/designs/STONE/instances_hier/inst2/instances_comb/g1
/designs/STONE/instances_hier/inst2/instances_comb/g2
/designs/STONE/instances_hier/inst2/instances_comb/g3
/designs/STONE/instances_hier/inst2/instances_comb/g4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all des outs

```
all des outs
     [-clock clock...] [-clock_domains clock_domain...]
     [design]
```

Generates a Tcl list of all output ports of the specified clock or clock domain.

Options and Arguments

`-clock` *`clock`*	Returns a list of output ports for the specified clock or clocks. This option is to find the ports with respect to a clock waveform, not a clock input port. It is valid only after you constrain the input and output ports.
`-clock_domains` *`clock_domain`*	Returns a list of output ports for the specified clock domain or domains.
`design`	Returns a list of output ports for the specified design. If a design is not specified, the output ports for the current design will be returned.

Example

- The following example returns all the output ports for the clock named `clock1`.

```
legacy_genus:/> all des outs -clock clock1
{/designs/STONE/ports_out/out1[1]} {/designs/STONE/ports_out/out1[0]}
{/designs/STONE/ports_out/out2[3]} {/designs/STONE/ports_out/out2[2]}
{/designs/STONE/ports_out/out2[1]} {/designs/STONE/ports_out/out2[0]}
/designs/STONE/ports_out/out3 /designs/STONE/ports_out/out4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all des seqs

```
all des seqs
     [-clock clock...] [-clock_domains clock_domain...]
     [-exclude instance...]
     [-level_sensitive | -edge_triggered]
     [-no_hierarchy]
     [-data_pins] [-output_pins] [-clock_pins]
     [-master_slave] [-slave_clock_pins]
     [-inverted_output] [design...]
```

Generates a Tcl list of all the flip-flops and latches in the design. Use the `-level_sensitive` option to return only the latches in the design.

Options and Arguments

`-clock` *`clock`*	Returns a list of sequential instances for the specified clock or clocks.
`-clock_domains` *`clock_domain`*	Returns a list of sequential instances for the specified clock domain or domains.
`-clock_pins`	Returns the clock pins in the design.
`-data_pins`	Only returns a list of data pins.
`design`	Returns a list of sequential instances for the specified design. If a design is not specified, the sequential instances for the current design will be returned.
`-edge_triggered`	Only returns a list of flip-flops in the design.
`-exclude` *`instance`*	Specifies a list of instances to be excluded.
`-inverted_output`	Returns sequential instances that have an inverted output (Qbar)
`-level_sensitive`	Only returns a list of latches in the design.
`-master_slave`	Returns the master slave flops.
`-no_hierarchy`	Returns sequential elements only at the top-level.
`-output_pins`	Returns the output pins in the design.
`-slave_clock_pins`	Returns the slave clock pins of master slave flops.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example returns all the sequential instances for the design named `REID`.

```
legacy_genus:/> all des seqs REID
{/designs/REID/instances_hier/accum1/instances_seq/r_reg[1]}
{/designs/REID/instances_hier/accum1/instances_seq/r_reg[2]}
{/designs/REID/instances_hier/accum1/instances_seq/r_reg[3]}
```

- The following example returns all the data pins for the design named `REID`.

```
legacy_genus:/> all des seqs REID -data_pins
{/designs/cpu/instances_hier/accum1/instances_seq/r_reg[1]/pins_in/d}
{/designs/cpu/instances_hier/accum1/instances_seq/r_reg[2]/pins_in/d}
{/designs/cpu/instances_hier/accum1/instances_seq/r_reg[3]/pins_in/d}
```

- The following example returns the master slave clock with the `-master_slave` option and then the slave clock pins of that master slave clock with the `-slave_clock_pins` option. If you are using the `map_to_master_slave_lssd` attribute, you must specify it before loading any libraries.

```
legacy_genus:/> set_attribute map_to_master_slave_lssd true
legacy_genus:/> set_attribute library jess.lib
...
legacy_genus:/> all des seqs -master_slave

/designs/summers/instances_seq/flop

legacy_genus:/> all des seqs -slave_clock_pins

/designs/summers/instances_seq/flop/pins_in/t0
```

Related Information

Related attributes: lssd_master_clock

map_to_master_slave_lssd

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all lib

```
all lib {bufs | ties}
```

Generates a Tcl list of library cell objects based on the specified object. For more information on specific `all lib` commands, see Related Information.

Options and Arguments

`bufs`	Generates a list of all the buffers in the loaded library.
`ties`	Generates a list of all the tie-cells in the loaded library.

Related Information

Related commands:	all lib bufs on page 556
	all lib ties on page 557

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all lib bufs

```
all lib bufs
```

Generates a Tcl list of all the buffers in the loaded library or libraries.

Example

- The following example returns all the buffers that were defined in the loaded library.

```
legacy_genus:/> all lib bufs
/libraries/slow/libcells/BUFX1 /libraries/slow/libcells/BUFX12
/libraries/slow/libcells/BUFX16 /libraries/slow/libcells/BUFX2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all lib ties

```
all lib ties {-lo | -hi | -hilo}
```

Generates a Tcl list of all the tie-cells in the loaded library.

Options and Arguments

`-hi`	Returns only a list of tie-hi cells (1 value tie cells).
`-hilo`	Returns only a list of tie-hi-lo cells (0 and 1 value tie cells).
`-lo`	Returns only a list of tie-lo cells (0 value tie cells).

Example

- The following example returns a list of all tie-hi cells in the library named `LUX`.

```
legacy_genus:/> all lib ties -hi
/libraries/LUX/libcells/TIEHI /libraries/LUX/libcells/ANTENNA
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

analyze_library_corners

```
analyze_library_corners
     {-libraries list | -cpf file}
     [-buffer_libcell libcell]
     [-fanout integer] [-fanin integer]
      [> file]
```

Reads in the specified multi-corner libraries and determines the slowest corner. Multi-corner libraries have the same libcells but are each characterized for a specific set of operating conditions resulting in different delay and slew values.

For each libcell the tool takes into account a given load at the input pins (specified through the number of buffers at the input) and a given load at the output pins (specified through the number of buffers at the output). Given that configuration, the tool calculates all timing arcs for the libcells for each corner and reports the average delay per corner. In addition, it reports the list of libcells whose delay exceeds the delay of the corresponding cell in the slowest library.

Important

This command should be the only command run in the synthesis session.

Options and Arguments

`-buffer_libcell` *`cell`*

Specifies the libcell to be used as buffer.

`-cpf` *`file`*

Specifies the name of the CPF file that has the libraries for the multi corners.

`-fanin` *`integer`*

Specifies the number of buffers to consider in the fanin of the input pins of each libcell.

Default: `2`

`-fanout` *`integer`*

Specifies the number of buffers to consider in the fanout of the output pins of each libcell

Default: `10`

`-libraries` *`list`*

Specifies the list of multi-corner libraries.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command reads in the libraries from the CPF file. There are 8 libraries. The report shows the average delay for each library (corner) and indicates that library `XS` in set4 has the largest delay.

```
analyze_library_corners -cpf test.cpf


             Library filename                Average delay
-------------------------------------------------------------
/libraries/library_domains/set8/MF           170
/libraries/library_domains/set7/MS           176
/libraries/library_domains/set6/XF           100
/libraries/library_domains/set4/XS           352
/libraries/library_domains/set2/F            162
/libraries/library_domains/set5/S            193
/libraries/library_domains/set3/VF           115
/libraries/library_domains/set1/VS           335


The slowest library : /libraries/library_domains/set4/XS
The average delay for this lib : 352

     Libcell              Library             Delay         Slowest Library          Delay
-----------------------------------------------------------------------------------------------
     AO22XA    /lib*/library_domains/set1/VS   354    /lib*/library_domains/set4/XS    346
-----------------------------------------------------------------------------------------------
    AND2CSXA   /lib*/library_domains/set1/VS   283    /lib*/library_domains/set4/XS    280
-----------------------------------------------------------------------------------------------
     BUFCSXA   /lib*/library_domains/set1/VS   241    /lib*/library_domains/set4/XS    239
-----------------------------------------------------------------------------------------------
    NAND3BXA   /lib*/library_domains/set1/VS   356    /lib*/library_domains/set4/XS    349
-----------------------------------------------------------------------------------------------
   NAND3BNXA   /lib*/library_domains/set1/VS   315    /lib*/library_domains/set4/XS    311
-----------------------------------------------------------------------------------------------
......
......
-----------------------------------------------------------------------------------------------
      BUFXA    /lib*/library_domains/set1/VS   241    /lib*/library_domains/set4/XS    239
-----------------------------------------------------------------------------------------------
     AND3XA    /lib*/library_domains/set1/VS   343    /lib*/library_domains/set4/XS    338
-----------------------------------------------------------------------------------------------
     OA22XA    /lib*/library_domains/set1/VS   348    /lib*/library_domains/set4/XS    340
-----------------------------------------------------------------------------------------------
```

Note: In the report, libraries was replaced with lib* to fit the report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_design

```
check_design [-lib_lef_consistency]
     [-undriven [-threshold_fanout integer]]
     [-multidriven] [-unloaded] [-unloaded_comb]
     [-unresolved] [-assigns] [-combo_loops]
     [-constant [-threshold_fanout] [-through_tie_cell]]
     [-preserved] [-logical_only] [-physical_only]
     [-only_user_hierarchy] [-vname]
     [-report_scan_pins | -skip_scan_pins] [-skip_async_pins]
     [-long_module_name] [-status] [-all] [design] [> file]
```

Provides information on undriven and multi-driven ports and pins, unloaded sequential elements and ports, unresolved references, constants connected ports and pins, any assign statements and preserved instances in the given design. In addition, the command can report any cells that are not in both .lib and the physical libraries (LEF files). By default, if you do not specify an option, the `check_design` command reports a summary table with this information.

Options and Arguments

`-all`	Reports all information for the design with a summary at the end.
`-assigns`	Reports assign statements in the design.
`-combo_loops`	Returns 1 if a combinational cycle is detected, otherwise returns 0.
`-constant`	Reports ports and pins in the design that are connected to a constant.
`design`	Specifies the name of the design to write the report. This option is required when multiple designs are loaded.
`file`	Specifies the name of the file to write the report.
`-lib_lef_consistency`	Reports the cells that are present in .lib but not in the LEF file(s) and vice versa.
`-logical_only`	Reports all instances of logical-only libcells, that is, library cells that are only available in a .lib library.
`-long_module_name`	Reports subdesigns whose name exceeds 1.5K.
`-multidriven`	Reports ports and pins in the design that are multi-driven.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-only_user_hierarchy`	Skips the tool generated internal hierarchies (such as Mux and Adders) when performing checks and only reports checks on user-created hierarchies.
`-physical_only`	Reports all instances of physical-only libcells, that is, library cells that are only available in a LEF library.
`-preserved`	Reports all hierarchical and leaf instances in the design for which the `preserve` attribute is set to `true`.
`-report_scan_pins`	Includes the scan (DFT-related) pins in the checks and reports.
`-skip_async_pins`	Excludes the asynchronous pins of flops and latches when reporting on undriven or unloaded pins.
`-skip_scan_pins`	Excludes the scan (DFT-related) pins from the checks and reports. **Note:** By default, the scan pins are included in all checks and reports.
`-status`	Returns 1 if any instances were reported, otherwise returns 0.
`-threshold_fanout`	Filters undriven or constant pins and ports with a fanout below the specified threshold. **Note:** This option must be specified with the `-constant` option.
`-through_tie_cell`	Reports pins and ports connected to a constant through tie high/low cells. **Note:** This option must be specified with the `-constant` option.
`-undriven`	Reports ports and pins in the design that are undriven.
`-unloaded`	Reports ports and sequential elements in the design that are unloaded.
`-unloaded_comb`	Reports the combinational elements in the design that are unloaded.
`-unresolved`	Reports unresolved references and empty modules in the design.
`-vname`	Uses the Verilog names instead of Genus design hierarchy path names in the report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example shows a sample report given when the command is executed without any options.

```
legacy_genus:/> check_design
  Checking the design.

         Check Design Report
         --------------------
 Summary
 -------
                Name                 Total
-------------------------------------------
Unresolved References                    0
Empty Modules                            0
Unloaded Port(s)                         0
Unloaded Sequential Pin(s)               0
Assigns                                  0
Undriven Port(s)                         0
Undriven Leaf Pin(s)                     0
Undriven hierarchical pin(s)             1
Multidriven Port(s)                      0
Multidriven Leaf Pin(s)                  0
Multidriven hierarchical Pin(s)          0
Multidriven unloaded net(s)              0
Constant Port(s)                         0
Constant Leaf Pin(s)                     0
Constant hierarchical Pin(s)             0
Preserved leaf instance(s)               5
Preserved hierarchical instance(s)       1
Libcells with no LEF cell              601
Physical (LEF) cells with no libcell   846

  Done Checking the design.
```

- The following example reports the preserved instances in your design.

```
legacy_genus:/> check_design -preserved
  Checking the design.

         Check Design Report
         --------------------

  Preserved instances(s)
  ----------------
design 'test1' has the following preserved combinational instance(s)
/designs/test1/instances_comb/and1
/designs/test1/instances_comb/and2
/designs/test1/instances_comb/and3
/designs/test1/instances_hier/U1/instances_comb/and4
Total number of preserved combinational instances in design 'test1' : 4
design 'test1' has the following preserved sequential instance(s)
/designs/test1/instances_seq/dff1
Total number of preserved sequential instances in design 'test1' : 1
design 'test1' has the following preserved hierarchical instance(s)
/designs/test1/instances_hier/U1
Total number of preserved hierarchical instances in design 'test1' : 1

  Done Checking the design.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports the cells which are only in .lib or in the physical library (LEF files).

```
legacy_genus:/> check_design -lib_lef_consistency
  Checking the design.

         Check Design Report
         --------------------

Libcells with no corresponding LEF
----------------------------------
/libraries/mylib.slow/libcells/ACCSHCINX2
...
/libraries/mylib.slow/libcells/p_SMDFFHQX8

Total number of cell(s) with only library (.lib) info : 601

LEF cells with no corresponding libcell
---------------------------------------
/libraries/physical_cells/libcells/AN2D0
...
/libraries/physical_cells/libcells/XOR4D4
Total number cell(s) with only physical (LEF) Info : 846

  Done Checking the design.
```

- The following command reports all undriven pins including the DFT-related pins (highlighted in the report below).

```
legacy_genus:/> check_design -undriven
  Checking the design.

         Check Design Report
        --------------------

 Undriven Port(s)/Pin(s)
 -------------------------
No undriven combinational pin in design 'test1'

The following sequential pin(s) in design 'test1' are undriven
/designs/test1/instances_seq/dff1/pins_in/SE
/designs/test1/instances_seq/dff1/pins_in/SI
Total number of sequential undriven pins in design 'test1' : 2

The following hierarchical pin(s) in design 'test1' are undriven
/designs/test1/instances_hier/U1/pins_in/A       (fanout : 0)
Total number of hierarchical undriven pins in design 'test1' : 1

No undriven port in 'test1'

  Done Checking the design.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports all undriven pins *excluding* the DFT-related pins.

```
legacy_genus:/> check_design -undriven -skip_scan_pins
  Checking the design.

         Check Design Report
        --------------------

 Undriven Port(s)/Pin(s)
 -----------------------
No undriven combinational pin in 'test1'

No undriven sequential pin in 'test1'

The following hierarchical pin(s) in design 'test1' are undriven
/designs/test1/instances_hier/U1/pins_in/A       (fanout : 0)
Total number of hierarchical undriven pins in design 'test1' : 1

No undriven port in 'test1'

  Done Checking the design.
```

- The following command reports physical-only instances.

```
legacy_genus:/> check_design -phys
  Checking the design.

         Check Design Report
         --------------------

  Physical only instances(s)
  ----------------
design 'rct' has the following physical only instance(s)
/designs/rct/instances_comb/ram_bank0
/designs/rct/instances_comb/ram_bank1
/designs/rct/instances_comb/ram_sort

  Done Checking the design.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_timing_intent

```
check_timing_intent
     [-mode mode]
     [-verbose]
```

Prints a timing lint report.

`check_timing_intent` checks the SDC quality and timing attributes placed on the current design and accordingly issues warnings as needed. These warnings do not indicate design has problems, instead indicate that the SDC has potential errors.

Note: It is strongly recommended to execute this command after `read_sdc` command, and fix these warnings before downstream run. You need to double check each warning to confirm whether to ignore or to fix it.

By default, this command reports verbose information for each warning.

The checks included are:

- Unconnected or logic driven clocks
- Sequential data pins driven by a clock signal
- Sequential clock pins without clock waveform
- Sequential clock pins with multiple clock waveforms
- Generated clocks without clock waveform
- Generated clocks with incompatible options
- Generated clocks with multi-master clock
- Paths constrained with different clocks
- Loop-breaking cells for combinational feedback
- Nets with multiple drivers
- Timing exceptions with no effect
- Suspicious multi_cycle exceptions
- Timing exceptions with no effect
- Suspicious multi_cycle exceptions
- Timing exceptions with no effect
- Suspicious multi_cycle exceptions

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Pins/ports with conflicting case constants
- Pins/ports with conflicting case constants
- Pins/ports with conflicting case constants
- Inputs without clocked external delays
- Outputs without clocked external delays
- Inputs without external driver/transition
- Outputs without external load
- Exceptions with invalid timing start or endpoints

Options and Arguments

`-verbose`	Prints a verbose report.
`-mode` *`mode`*	Specifies that the report should only apply to the specified *`mode`*.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_ports

`clock_ports [design]`

Returns input ports of your design that are clock inputs.

Note: Only input ports at the top level are listed. Gated clocks and clock pins that are present in the hierarchical design internally (typical PLL outputs) will not be identified.

Note: This command is useful when you are working with an unfamiliar design.

Options and Arguments

design	Specifies the design for which you want to list the clock input ports.

Examples

- The following example finds all of the clock ports of a design.

```
legacy_genus:/> clock_ports
/designs/alu/ports_in/clock
```

- In the following example, the `clock_ports` command is embedded within a `define_clock` command to apply a clock waveform to all clock input ports of the design.

```
legacy_genus:/> define_clock -period 3000 -name clock1 [clock_ports]
```

Related Information

Affected by this command: define_clock on page 456

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

compare_sdc

```
compare_sdc [-design string] [-rtl | -netlist file]
     -golden_ sdc files [-revised_sdc files]
     [-logfile file] [-detail] [> file]
```

Compares two (or two sets of) SDC files for a design and generates a report containing differences.

To run this command you need to have access to the Conformal ® Constraint Designer (CCD) software.

Options and Arguments

`-design` *`string`*	Specifies the top-level design in Genus.
`-detail`	Requests a detailed comparison report.
`file`	Specifies the file to which the report must be written.
`-golden_sdc` *`files`*	Specifies the UNIX path to the golden (original) SDC files.
`-logfile` *`file`*	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
`-netlist` *`file`*	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
`-revised_sdc` *`files`*	Specifies the UNIX path to the revised SDC files. If this option is not specified, the tool internally generates an SDC file for the current state of the design and uses this file for the comparison.
`-rtl`	Specifies to use the RTL as input.

Related Information

Related command: write_do_ccd compare_sdc on page 375

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_timing_bin

```
create_timing_bin -name string
     [-parent string]
     [-num_paths integer] [-worst integer]
     [-slack_limit delay] [-mode mode]
     [-from {instance|external_delay|clock|port|pin}...]
     [-through {instance|port|pin}..]
     [-to {instance|external_delay|clock|port|pin}...]
     [-exceptions exception...] [-cost_group cost_group...]
     [-paths string]
```

Allows you to select and save timing paths and their associated timing information, into a timing bin. Timing bins enable you to subsequently examine and analyze the timing paths without re-invoking the timer.

The data in a timing bin is a static snapshot created when the `create_timing_bin` command was invoked. If the design or constraints change, you can create new timing bin(s) and delete the old one(s).

Timing bin information is located at
`/designs/design/analysis/`*`timing_bin_name`* and
`/designs/design/analysis/`*`timing_bin_name`*`/sub_bins/`*`sub_bin_name`*

Timing path information for parent timing bins is located at
`/designs/design/analysis/`*`timing_bin_name`*`/paths/`*`path`*

Timing path information for sub-bins is located at
`/designs/design/analysis/`*`timing_bin_name`*`/sub_bins/`*`sub_bin_name`*`/paths/`*`path`*

Clock ports can be identified by the `is_clock` attribute present on the port.

Options and Arguments

`-cost_group` *`cost_group`*

Selects only paths for the specified cost groups.

`-exceptions`

Selects only paths to which one of the specified exceptions applies.

Timing exceptions are specified through one of the following Genus commands (or their SDC equivalent): `multi_cycle`, `path_adjust`, `path_delay`, `path_disable`, or `path_group`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This option can be combined with `-from`, `-through`, `-to` options to further restrict the path selection.

`-from {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, a combination of these, instances, or input ports to which specified external delay timing exceptions apply.

`-mode mode`

Specifies that the paths in the timing bin only apply for the specified mode.

`-name string`

Specifies the name of the timing bin to be created.

`-num_paths integer`

Specifies the maximum number of paths to include in the timing bin.

`-parent string`

Specifies the name of the timing bin from which you want to derive this bin.

`-paths string`

Includes the selected timing restricted paths. Create the string argument using the specify_paths command.

`-slack_limit delay`

Includes only paths with a slack less than the specified value (in picoseconds).

`-through {pin | port | instance}`

Specifies a Tcl list of points that a path must traverse. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-worst integer`

Specifies the number of worst paths to each endpoint to include.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Creates these attributes	Timing Bin Attributes
	Timing Path Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

fanin

```
fanin {pin | pgpin | port | subport | fnet | net}...
     [-min_logic_depth integer]
     [-max_logic_depth integer]
     [-min_pin_depth integer] [-max_pin_depth integer]
     [-structural [-stop_at_seqs] [-timing_model_comb]
       [-floating_hport] [-ignore_broken_timing_paths] ]
     [-startpoints] [-gui] [-vname]
```

Returns pins and ports in the fanin cone for the specified objects (pins, ports, or nets) starting from the nearest timing startpoints based on the timing arcs of the libcells of the instances. Use the `-structural` option to do a connectivity trace (ignoring timing arcs and functions) with every output of an instance having a “structural connection” to every input and vice-versa.

Options and Arguments

`-floating_subports`
: Reports only floating (unconnected) subports found in the connectivity-based structural trace. You can only specify this option with the `-structural` option.

  By default only non-floating subports and constants are returned.

`-gui`
: Highlights the instances along the fanin cone in the Physical Viewer. The instance at the start of the fanin cone is marked in yellow, the other instances are marked in red.

  Note: The GUI should be enabled to have any effect.

`-ignore_broken_timing_paths`
: Ignores broken timing paths during a connectivity-based structural trace. You can only specify this option with the `-structural` option.

`-max_logic_depth` *`integer`*
: Specifies the maximum number of logic levels to go back to report the fanin cone information.

  Default: Infinity

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max_pin_depth` *`integer`*

Specifies the maximum number of pin levels to go back to report the fanin cone information.

Default: Infinity

`-min_logic_depth` *`integer`*

Specifies the minimum number of logic levels to go back to report the fanin cone information.

Default: `0`

`-min_pin_depth` *`integer`*

Specifies the minimum number of pin levels to go back to report the fanin cone information.

Default: `0`

`{`*`pin`* `|` *`pgpin`* `|` *`port`* `|` *`subport`* `|` *`fnet`*`|` *`net`*`}`

Specifies the name of a pin, pgpin, port, subport, or net for which you want the fanin cone information.

`-startpoints` Returns only timing start points in the fanin cone.

`-stop_at_seqs` Stops the structural trace at sequential elements. You can only specify this option with the `-structural` option.

`-structural` Specifies to perform a connectivity-based structural trace instead of the default timing trace based on timing arcs.

Note: If there are missing timing arcs, for example, when using the SDC `set_case_analysis` command, the traces may report different results.

Tip

Use this option with care as it can increase runtime considerably.

`-timing_model_comb` Specifies to trace through combinational timing model libcells. You can only specify this option with the `-structural` option.

`-vname` Specifies to return the path to the pins and ports in Verilog style.

Note: This option does not apply to subports and constants.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- Consider the design below.

  The following example returns all the pins in the fanin cone for port `out`:

```
legacy_genus:/> fanin out
/designs/test/instances_seq/out_reg/pins_out/Q /designs/test/instances_seq/
out_reg/pins_in/CK
```

- The following example specifies to only return the start point for port `out` shown in the design above.

```
legacy_genus:/> fanin -startpoints out
/designs/test/instances_seq/out_reg/pins_in/CK
```

- The following example executes a path disable from all the start points that fan out to `reg1/D`.

```
legacy_genus:/> path_disable -from [fanin -startpoint reg1/D]
```

- The following example queries the fanin of the output pin `S` of the combinational instance adder shown in Figure 11-1 on page 575.

```
legacy_genus:/> fanin -startpoints S
```

  In this case, the command returns input port `IN` and clock pin `CK`

- Use the `ls -dir` command to format the output.

```
legacy_genus:/designs/malexander/ports_in> ls -dir [fanin in1[0]]
/designs/malexander/instances_hier/inst1/instances_comb/g43/pins_in/A
/designs/malexander/instances_hier/inst1/instances_comb/g43/pins_out/Y
/designs/malexander/ports_out/out1[1]
/designs/malexander/ports_out/out3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Use the `ls -dir` command with the redirect arrow to redirect the output to the specified file.

  ```
  legacy_genus:/designs/malexander/ports_in> ls -dir [fanin in1[0]] >
  project.txt
  ```

  You can also use the append arrows (">>").

- The following example queries the fanin of the output pin S of the combinational instance shown in Figure 11-1, but without using the `-startpoint` option.

  ```
  legacy_genus:/> fanin S
  ```

  In this case, the command returns in addition to the input port `IN` and clock pin `CK`, pins `A` and `Q`.

Figure 11-1 Example Design for fanin

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

fanout

```
fanout {pin | pgpin | port | subport | net}...
     [-min_logic_depth integer]
     [-max_logic_depth integer]
     [-min_pin_depth integer] [-max_pin_depth integer]
     [-structural [-stop_at_seqs] [-timing_model_comb]
       [-floating_subport] [-ignore_broken_timing_paths] ]
     [-endpoints] [-gui] [-vname]
```

Returns pins and ports in the fanout cone for the specified objects (pins, ports, or nets) stopping at the nearest timing endpoints based on the timing arcs of the libcells of the instances. Use the `-structural` option to do a connectivity trace (ignoring timing arcs and functions) with every output of an instance having a “structural connection” to every input and vice-versa.

Options and Arguments

`-endpoints`
: Returns only timing end points in the fanout cone.

`-floating_subports`
: Reports only floating (unconnected) subports found in the connectivity-based structural trace. You can only specify this option with the `-structural` option.

  By default only non-floating subports and constants are returned.

`-gui`
: Highlights the instances along the fanout cone in the Physical Viewer. The instance at the start of the fanout cone is marked in yellow, the other instances are marked in red.

  Note: The GUI should be enabled to have any effect.

`-ignore_broken_timing_paths`
: Ignores broken timing paths during a connectivity-based structural trace. You can only specify this option with the `-structural` option.

`-max_logic_depth` *`integer`*
: Specifies the maximum number of logic levels to go back to report the fanout cone information.

  Default: Infinity

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max_pin_depth` *`integer`*

Specifies the maximum number of pin levels to go back to report the fanout cone information.

Default: Infinity

`-min_logic_depth` *`integer`*

Specifies the minimum number of logic levels to go back to report the fanout cone information.

Default: `0`

`-min_pin_depth` *`integer`*

Specifies the minimum number of pin levels to go back to report the fanout cone information.

Default: `0`

`{`*`pin`* `|` *`pgpin`* `|` *`port`* `|` *`subport`* `|` *`net`*`}`

Specifies the name of a pin, pgpin, port, subport, or net for which you want the fanout cone information.

`-stop_at_seqs`

Stops the structural trace at sequential elements. You can only specify this option with the `-structural` option.

`-structural`

Specifies to perform a connectivity-based structural trace instead of the default timing trace based on timing arcs.

Note: If there are missing timing arcs, for example, when using the SDC `set_case_analysis` command, the traces may report different results.

Tip

Use this option with care as it can increase runtime considerably.

`-timing_model_comb`

Specifies to trace through combinational timing model libcells. You can only specify this option with the `-structural` option.

`-vname`

Specifies to return the path to the pins and ports in Verilog style.

Note: This option does not apply to subports and constants.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example returns the pins in the fanout cone of port `en` in the design below.

```
legacy_genus:/> fanout en
/designs/test/instances_comb/g169/pins_out/Z /designs/test/instances_seq/
var2_reg/pins_in/D /designs/test/instances_comb/g170/pins_in/SL /designs/
test/instances_comb/g170/pins_out/Z /designs/test/instances_seq/var3_reg/
pins_in/D
```

- The following example executes a path disable on all the endpoints to which `reg1/CK` fans out.

```
legacy_genus:/> path_disable -to [fanout -endpoints reg1/CK]
```

- The following example returns all pins in the fanout cone up to two logic levels forward from the specified pin.

```
legacy_genus:/> fanout -max_logic_depth 2 B
/designs/top/instances_hier/m1/instances_comb/g2/pins_in/in_0
```

- Use the `ls -dir` command to format the output.

```
legacy_genus:/designs/malexander/ports_in> ls -dir [fanout in1[0]]
/designs/malexander/instances_hier/inst1/instances_comb/g43/pins_in/A
/designs/malexander/instances_hier/inst1/instances_comb/g43/pins_out/Y
/designs/malexander/ports_out/out1[1]
/designs/malexander/ports_out/out3
```

- Use `ls -dir` with the redirect arrow to redirect the output to the specified file.

```
legacy_genus:/designs/malexander/ports_in> ls -dir [fanout in1[0]] >
malexander.txt
```

You can also append arrows (">>").

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report

```
report {area | boundary_opto | case_analysis
     | cdn_loop_breaker  | cell_delay_calculation
     | clock_gating | clocks
     | congestion | cwd | datapath | design_rules
     | dft_chains | dft_clock_domain_info
     | dft_core_wrapper | dft_registers | dft_setup
     | dft_violations | disabled_transparent_latches
     | gates | hierarchy | instance | low_power_cells
     | low_power_intent | memory | memory_cells | messages
     | mode | multibit_inferencing
     | net_cap_calculation | net_delay_calculation
     | net_res_calculation | nets
     | ple | port | power | power_domain | proto | qor
     | scan_compressibility | sequential
     | slew_calculation | state_retention | summary
     | test_power | timing | units | utilization | yield}
```

Generates the specified report on synthesis results. For more information, see the Related Information.

All reports have a header which contains information about the version of the tool used to generate the report, the time that the report was generated, the module for which the report is generated, the technology libraries used for synthesis, the operating conditions. The next two lines in the header depend on the setting of some attributes.

In addition, if you are running synthesis using wireload models (`interconnect_mode` attribute set to `wireload`), the header has a `Wireload mode` entry that corresponds to the value of the `wireload_mode` root attribute. In this case, Genus uses the cell area specified in the timing libraries. Consequently, the `Area mode` entry is set to `timing library`.

If you are running any of the physical synthesis flows (`interconnect_mode` attribute set to `ple`), the header has an `Interconnect mode` entry which can have the following values:

- `global`—report generated before synthesis is run (all physical flows) or generated after the design is synthesized without placement information (simple PLE flow)
- `spatial`—report generated after the design is synthesized using a fast placement (spatial flow).
- `placement`—report generated after the design is synthesized using detailed placement information (Genus-Physical flow)

When running a physical flow, Genus uses the cell area specified in the LEF libraries and therefore `Area mode` is set to `physical library`.

You can automatically write out a gzip compressed report file. For example:

```
report port sample.gz
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: The `report memory` command does not support the gzip compressed output.

Options and Arguments

`area`	Reports the area of the synthesized and mapped design.
`boundary_opto`	Reports a summary of the boundary optimization.
`case_analysis`	Reports the user-set case constants on pins and ports.
`cdn_loop_breaker`	Reports the loop breaker buffers added by the tool and breaks the combinational loops for timing analysis.
`cell_delay_calculation`	Reports how the cell delay of a libcell instance is calculated.
`clock_gating`	Reports clock-gating information for the design.
`clocks`	Generates a report on the clocks of the current design.
`congestion`	Reports the congestion summary.
`cwd`	Generates a ChipWare Developer report.
`datapath`	Reports datapath operators that were inferred from the design.
`design_rules`	Reports the design rule violations.
`dft_chains`	Reports the scan chain data for the design.
`dft_clock_domain_info`	Reports the DFT clock domain information.
`dft_core_wrapper`	Reports the wrapper cells inserted in the design.
`dft_registers`	Reports the scan status of all flip-flops in the design.
`dft_setup`	Reports the DFT setup information.
`dft_violations`	Reports the DFT violations.
`disabled_transparent_latches`	Reports disabled transparent latches.
`gates`	Generates a gates report.
`hierarchy`	Reports the design hierarchy information.
`instance`	Generates a report on the instances of the current design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`low_power_cells`	Reports information about the low power cells inserted in the design.
`low_power_intent`	Prints out the power intent included in the power intent file that was read in.
`memory`	Reports the memory usage in the compilation environment.
`memory_cells`	Reports the memory cells in the library.
`messages`	Generates a summary of error messages that have been issued.
`mode`	Reports the active and inactive modes of an instance.
`multibit_inferencing`	Reports on multibit cells in the design or library.
`net_cap_calculation`	Reports how the capacitance of the net is calculated.
`net_delay_calculation`	Reports how the net delay is calculated.
`net_res_calculation`	Reports how the resistance of the net is calculated.
`nets`	Generates a report on the nets of the current design.
`opcg_equivalents`	Reports the `OPCG`-equivalence mappings.
`ple`	Reports physical layout estimation data
`port`	Generates reports on the ports of the current design.
`power`	Generates a power leakage report.
`power_domain`	Generates a report with power domain information.
`proto`	Generates a report with prototype synthesis information.
`qor`	Generates a QoR report.
`scan_compressibility`	Reports the scan compressibility of a design.
`sequential`	Generates a report on the sequential elements of the design.
`slew_calculation`	Reports how the slew on the driver pin of a libcell instance is calculated.
`summary`	Generates an area, timing, and design rule violations report.
`test_power`	Reports estimated power of design during scan test

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`timing`	Generates a timing report.
`units`	Reports the units used in the libraries.
`utilization`	Reports the floorplan utilization for the design.
`yield`	Generates a yield report.

Related Information

Related commands:

report area on page 584
report boundary_opto on page 587
report case_analysis on page 589
report cdn_loop_breaker on page 590
report cell_delay_calculation on page 592
report clock_gating on page 593
report clocks on page 600
report congestion on page 602
report datapath on page 603
report design_rules on page 608
report dft_chains on page 610
report dft_clock_domain_info on page 615
report dft_core_wrapper on page 617
report dft_registers on page 620
report dft_setup on page 624
report dft_violations on page 629
report disabled_transparent_latches on page 632
report gates on page 633
report instance on page 639
report memory on page 642
report memory_cells on page 643

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report messages on page 644
report mode on page 648
report multibit_inferencing on page 650
report net_cap_calculation on page 657
report net_delay_calculation on page 658
report net_res_calculation on page 660
report nets on page 661
report opcg_equivalents on page 664
report ple on page 666
report port on page 668
report power on page 669
report power_intent on page 681
report power_intent_instances on page 685
report proto on page 688
report qor on page 690
report scan_compressibility on page 697
report sequential on page 699
report slew_calculation on page 702
report summary on page 703
report test_power on page 705
report timing on page 709
report units on page 721
report utilization on page 722
report yield on page 723

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report area

```
report area
     [-depth integer [-hinst instance]]
     [-min_count integer] [-physical] [-gates]
     [-normalize_with_gate lib_cell
     [-show_leaf_cells] [-summary] [-detail]
     [-skip_modules_logical_hier_false]
     [design]... [> file]
```

Reports the following information:

- The total count of cells mapped against the hierarchical blocks in the current design
- The combined cell area in each of the blocks and the top level design (hierarchical breakup)

  The `Cell Area` numbers are based on the cell implementations taken from the technology library after mapping. However, in PLE mode, the numbers are based on the information in the LEF libraries. It might be 0 if the information is missing in the LEF libraries.
- The `Net Area` refers to the estimated post-route net area and is only meaningful if you read in the LEF libraries. Net area is based on the minimum wire widths defined in the LEF and capacitance table files and the area of the design blocks.
- The wireload model adopted for each of the blocks

Note: The units used in the report depend on the units used in the library.

Options and Arguments

`-depth` *`integer`*	Specifies the number of levels of recursion.
`design`	Specifies the design for which you want to generate a report. You can also `cd` into the particular design directory and generate the report.
`-detail`	Reports detailed area information. Reports area for different categories, such as, combinational cells, sequential cells, buffers, inverters, clock-gating instances, and so on.
`file`	Specifies the name of the file to which to write the report.
`-gates`	Prints the area of all combinational cells, sequential cells, and hierarchical instances

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hinst` *`instance`* Reports the area of the leaf instances in the specified hierarchical instance. This option must be specified together with the `-depth` option.

`-min_count` *`integer`* Specifies the minimum instance count per block.

`-normalize_with_gate` *`lib_cell`*

Normalizes all area numbers in the report with the area of the specified library cell.

`-physical` Specifies to use the LEF cell width and height values to calculate the area.

`-show_leaf_cells` Prints the leaf instances at the design level in the report.

`-skip_modules_logical_hier_false`

Skips reporting those modules for which the `logical_hier` attribute has the `false` value.

`-summary` Prints the area summary for the design.

Examples

- The following example generates the area report for the current design.

```
legacy_genus:/> report area
====================================================================
  Generated by:           version
  Generated on:          Current date Current time
  Module:                 mult_bit_muxed_add
  Technology library:     tutorial
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

*************  Area  *************

     Instance       Cells  Cell Area  Net Area     Wireload
-------------------------------------------------------------------
mult_bit_muxed_add      6         69         0  suggested_10K (S)
  ma1                   3         35         0  suggested_10K (S)
  ma0                   3         35         0  suggested_10K (S)

 (S) = wireload was automatically selected
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command prints the area summary.

```
legacy_genus:/> report area -summary
============================================================
...
============================================================

     Instance       Cells  Cell Area  Net Area     Wireload
-----------------------------------------------------------------
mult_bit_muxed_add      6         69         0  suggested_10K (S)
```

- The following command reports the area for the leaf instances of instance `ma1`.

```
legacy_genus:/> report area -hinst ma1 -depth 2
============================================================
...
============================================================

Instance  Cells  Cell Area  Net Area     Wireload
-------------------------------------------------------
  ma1         3         35         0  suggested_10K (S)
    g53       0         12         0  suggested_10K (S)
    g52       0         12         0  suggested_10K (S)
    g51       0         10         0  suggested_10K (S)

 (S) = wireload was automatically selected
```

Related Information

Affected by these commands:	syn_generic on page 526
	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	logical_hier
	shrink_factor

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report boundary_opto

`report boundary_opto [instance]... [-hierarchy instance]`

Reports a summary of the boundary optimization done on the design.

This is a summary of boundary changes on the hierarchical pins.

Options and Arguments

`-hierarchy` *`instance`*	Reports the boundary optimization done on all instances below the specified instance.
`instance`	Reports the boundary optimization done on the specified instance.

Example

Consider the following input RTL.

```
module top(in1,in2,out,out1,out2);
    input in1,in2;
    output out,out1,out2;
    child inst(in1,in2,out,out1,out2);
endmodule

module child(a,b,out,out1,out2);
    input a,b;
    output out,out1,out2;
    and u1(n_1,b,a);
    assign out = n_1;
    assign out1 = ~a;
    assign out2 = ~n_1;
endmodule
```

After the `synthesize -to_map` command, report the boundary optimization.

```
legacy_genus:/> report boundary_opto
============================================================
  ...
============================================================

Instance  Pin                     Boundary Change
-------------------------------------------------------------------
inst     out1  routed opposite signal through 'inst/a' around 'inst'
         out2  pushed opposite signal through 'inst/out'
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related attribute: boundary_change

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report case_analysis

```
report case_analysis
     design [> file]
```

Reports ports and pins that have a `set_case_analysis` constraint.

This command generates a User Case Analysis table, which reports the ports and pins that have a `set_case_analysis` constraint. These ports and pins are constant only for timing analysis.

You can optionally generate a Logic Constant Value table, which reports all the ports and pins that have a constant status as part of the design. Information from this table includes ports or pins with `set_logic constraints` (`set_logic_zero` and `set_logic_one`), netlist explicit constants, and output pins with constant output functions.

Options and Arguments

design	Specifies the design for which to generate the report.
file	Reports the case constants.

Example

```
 ------------------------------
  report case_analysis
 ------------------------------
Pin Name Case
--------------
 OR1/A      0
 en         1
```

Related Information

Set by this command:	set_case_analysis on page 1531
Set by these attributes:	(pin) timing_case_logic_value
	(port) timing_case_logic_value

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report cdn_loop_breaker

```
report cdn_loop_breaker [-sdcfile string]
     [-version string] [design] [> file]
```

Reports the loop breaker buffers that were added by the timing engine to break the combinational loops during timing analysis.

The report lists for every loop breaker the instance name, and the driver and the load of the loop breaker.

Options and Arguments

design	Specifies the design for which you want the report.
file	Redirects the report to the specified file.
-sdcfile *string*	Creates an SDC file with the appropriate `set_disable_timing` and `set_false_path` settings.
-version *string*	Specifies a particular SDC version for the SDC file created with the `-sdcfile` option. The available versions are: `1.1, 1.3, 1.4, 1.5` or `1.7.` *Default*: `1.7`.

Example

The following report shows the loop breakers inserted in design `loop`.

```
legacy_genus:/> report cdn_loop_breaker
============================================================
  Generated by:           version
  Generated on:           date
  Module:                 loop
  Technology library:     tutorial 1.1
  Operating conditions:   typical_case (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

   CDN Loop breaker      Driver   Load
----------------------------------------
inst1/cdn_loop_breaker inst1/i1/Y i0/B
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: remove_cdn_loop_breaker on page 1351

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report cell_delay_calculation

```
report cell_delay_calculation
     -from_pin pin  -to_pin pin
     [-from_rise] [-from_fall]
     [-to_rise] [-to_fall]
     [-mode mode] [> file]
```

Reports how the cell delay is calculated from the look up table in the loaded technology library. Specify the cell by choosing its the driving and loading pins.The formula for calculating the delay is provided at the bottom of the report.

Options and Arguments

file	Redirects the report to the specified file.
-from_pin *pin*	Specifies the driving pin.
-mode *mode*	Specifies that the report should only apply to the specified *mode*.
-from_fall	Uses the fall delay calculation from the driving pin.
-from_rise	Uses the rise delay calculation from the driving pin.
-to_fall	Uses the fall delay calculation of the loading pin.
-to_pin *pin*	Specifies the loading pin.
-to_rise	Uses the rise delay calculation of the loading pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report clock_gating

```
report clock_gating [-instance hier_instance]
     [ [-gated_ff] [-ungated_ff] [-no_hierarchical] [-detail]
       [-fanout_histogram [-step {{start stop}...}]]
     | {-clock clock_list |-clock_pin {pin|port|subport}...}
       [-detail]
     | -cg_instance cg_instance...
     | [-multi_stage] [-multi_stage_count_from_flop]
       [-multi_stage_count_from_root]]
     [-multi_stage_tree_view] [-include_one_flop]
     [-include_all_flops]
     [design] [> file]
```

Reports clock-gating information for the design.

If you specify this command without any options, the command prints a summary report of the clock gating inserted in the design that includes the number of RC and non-RC clock-gating instances, the number of RC and non-RC gated flip-flops with the average toggle saving (in percent), the number of ungated flip-flops, and the total number of flip-flops in the design.

Note: The first two lines refer to the *leaf* clock-gating instances (RC and non-RC) that were added. If multi-stage clock gating is present, two more lines are added to the top of the summary reporting the multi-stage clock gating instances (RC and non-RC).

The return value of this command is the total number of clock-gating logic inserted in the design.

Note: After importing third-party clock-gating logic, this clock-gating logic will be reported as “RC Clock Gating Instances.”

Options and Arguments

`-cg_instance` *cg_instance*

Reports detailed clock-gating information for the specified clock-gating instances. Information includes the library cell used for the clock-gating cell, the clock-gating style, the signals connected to the inputs and outputs of the gating logic, and the flip-flops gated by this gating cell.

A clock-gating instance is the hierarchical instance with the clock-gating logic inside.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-clock` *`clock_list`*
: Limits the report to the specified clocks. The specified clocks must that have been defined through either the `define_clock` command or through the SDC constraints.

`-clock_pin {`*`pin`* `|` *`port`*`|` *`subport`*`}`
: Limits the report to the specified clock pins. Use this option if you did not define the clocks. You can specify clock pins, clock ports or clock subports.

  Note: If both `-clock` and `-clock_pin` options are specified, the `-clock` option takes precedence.

`design`
: Specifies the design for which you want to generate the report.

`-detail`
: Reports detailed clock-gating information. Lists all the clock-gating instances inserted, including the library cell used for the clock-gating cell, the clock-gating style, the signals connected to the inputs and outputs of the gating logic, and the toggle reduction achieved. In addition, it lists for each clock-gating instance the names of the gated flip-flops with the register efficiency.

  If you specify only this option, the return value of this command is the total number of clock-gating logic inserted in the design.

`-fanout_histogram`
: Prints for a set of fanout ranges, the number of clock-gating instances with a fanout in that range, and the total number of flip-flops that these clock-gating instances gate.

`file`
: Specifies the name of the file to which to write the report.

`-gated_ff`
: Reports all the flip-flops that are clock gated and the clock-gating instances that gate the flip-flops. In addition, it lists for all gated flip-flops the register efficiency.

  If you specify only this option, the return value of this command is the total number of flip-flops that are gated in the design.

`-include_all_flops`
: Specifies to include all flops per CG when reporting multi-stage CG hierarchy in a tree view.

`-include_one_flop`
: Specifies to include one flop per CG when reporting multi-stage CG hierarchy in a tree view.

`-instance` *`hier_instance`*
: Prints the clock-gating report for the specified hierarchical instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-multi_stage`

Shows the clock-gating instance hierarchy. This option can only be combined with other `-multi_stage` options.

This option and the `-multi_stage_count_from_flop` option are equivalent.

`-multi_stage_count_from_flop`

Counts the levels (stages) of the multi-stage clock-gating instances starting from the flop. The level or stage of the clock-gating instances driving the flops will be 0.

`-multi_stage_count_from_root`

Counts the levels (stages) of the multi-stage clock-gating instances starting from the root or the top-level clock-gating instance. The level or stage of the top-level clock-gating instances will be 1.

In this case, the stages will correspond to the values of the `lp_clock_gating_stage` attributes of the clock-gating instances.

`-multi_stage_tree_view`

Generates the report of multi-stage CG hierarchy in a tree view.

`-no_hierarchical`

Limits the clock-gating information to the current module.

By default, the report command traverses the hierarchy starting from the current module and reports all the clock gating found in the current module and its children modules.

`-step` **{{***start stop***}...}**

Specifies user-defined ranges for the fanout histogram. Specify a list of lists. Each list defines a range. The stop point of the range cannot be smaller than the start point. In addition, the start point of each range must be larger than the stop point of the previous range.

The tool will report on the defined ranges and the uncovered ranges.

Note: This option can only be specified with the `-fanout_histogram` option.

`-ungated_ff`

Reports all the flip-flops that are not clock gated, and lists whether the flop was excluded for clock-gating or not.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you specify only this option, the return value of this command is the total number of flip-flops that are not gated in the design.

When you specify this option with the `-detail` option, the report lists the specific reason why the flip-flops were not gated.

Examples

The following reports show output generated after clock gating has been inserted.

- The following command reports all the flip-flops that are clock gated. In this case, the return value of the command is 8.

```
legacy_genus:/> report clock_gating -gated_ff
============================================================
...
============================================================

Gated Flip-flops
----------------
 Module  Clock Gating Instance  Origin  Fanout  Gated Flip-flops  Register Efficiency
-----------------------------------------------------------------------------------------
 alu     RC_CG_HIER_INST0         RC       8    aluout_reg[0]    100.000
                                               aluout_reg[1]    100.000
 ...
                                               aluout_reg[6]    100.000
                                               aluout_reg[7]    100.000
-----------------------------------------------------------------------------------------
 Total   1                                8
-----------------------------------------------------------------------------------------


8
```

- The following command gives the reason why the flip-flops remain ungated. In this case, there is no return value.

```
legacy_genus:/> report clock_gating -ungated -detail
==============================================================
...
==============================================================

Ungated Flip-flops With Detail
------------------------------

 Flops offering no power saving : 4
    /designs/test/instances_hier/my_ff/instances_seq/q_reg[0]
    /designs/test/instances_hier/my_ff/instances_seq/q_reg[2]
    /designs/test/instances_hier/my_ff/instances_seq/q_reg[3]
    /designs/test/instances_hier/my_ff/instances_seq/q_reg[1]
```

- The following command shows the fanout histogram.

```
legacy_genus:/> report clock_gating -fanout_histogram
==============================================================
...
==============================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
CG Fanout Histogram
-------------------
   Fanout-Size          Num-CGs         Total-FFs
-------------------------------------------------------------------------------
 1 to 3                    0            0
 4 to 15                   2            16
 16 to 63                  0            0
 64 to 255                 0            0
 256 and higher            0            0
```

- The following command shows the fanout histogram with user-defined ranges.

```
legacy_genus:/> report clock_gating -fanout_histogram -step {{1 6} {7 15}}
============================================================
...
============================================================

CG Fanout Histogram
-------------------
   Fanout-Size          Num-CGs         Total-FFs
-------------------------------------------------------------------------------
 1 to 6                    0            0
 7 to 15                   2            16
 16 and higher             0            0
```

- The following command generates detailed clock-gating information for the specified clock-gating instance.

```
legacy_genus:/> report clock_gating -cg_instance [find / -inst
RC_CG_HIER_INST0]
============================================================
...
============================================================

Detail
------
Clock Gating Instance : RC_CG_HIER_INST0
---------------------
  Origin:                Inserted by RC
  Libcell:               TLATNTSCAX2 (slow)
  Style:                 latch_posedge_precontrol
  Module:                alu (alu)
  Inputs:
    ck_in        =       clock (/designs/alu/ports_in/clock)
                         TCF = (0.75000, 0.571429/ns)
    enable       =       ena (/designs/alu/ports_in/ena)
                         TCF = (0.50000, 0.114583/ns)
    test         =       LOGIC0
  Outputs:
    ck_out       =       rc_gclk
                         TCF = (0.40890, 0.317708/ns)
  Toggle Reduction  =  44.40
  Gated FFs:
 Module  Clock Gating Instance  Origin  Fanout  Gated Flip-flops  Register Efficiency
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
-------------------------------------------------------------------------------
 alu     RC_CG_HIER_INST0         RC       8    aluout_reg[0]    100.000
                                                aluout_reg[1]    100.000
 ...
                                                aluout_reg[6]    100.000
                                                aluout_reg[7]    100.000
-------------------------------------------------------------------------------
 Total   1                                 8

1
```

- The following command reports the number of flip-flops that are clock gated by the specified clock.

```
legacy_genus:/> report clock_gating -clock [find / -clock clock]
Multi-stage clock gating for '/designs/alu/ports_in/clock'
=====================================
Max stage:    1
Total FFs with 0 stage of CG:  1
Total FFs with 1 stage of CG:  8
=====================================
Total FF:  9
```

- The following examples show a report of multi-stage clock-gating using the two different options to count the levels or stages.

```
legacy_genus:/> report clock_gating -multi_stage_count_from_flop
...
-------------------------------------------
        CG Instance         Level  Fanouts
-------------------------------------------
 RC_CG_SHARED_HIER_L1_INST      1        2
   RC_CG_HIER_INST1             0        4
   RC_CG_HIER_INST2             0        4
-------------------------------------------
legacy_genus:/> report clock_gating -multi_stage_count_from_root
...
-------------------------------------------------
        CG Instance         Level  Fanouts
-------------------------------------------------
 RC_CG_SHARED_HIER_L1_INST      1        2
   RC_CG_HIER_INST1             2        4
   RC_CG_HIER_INST2             2        4
-------------------------------------------------
```

Related Information

Affected by these commands:	syn_generic on page 526
	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	lp_clock_gating_add_obs_port

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

lp_clock_gating_add_reset

lp_clock_gating_cell

lp_clock_gating_control_point

lp_clock_gating_exclude

lp_clock_gating_style

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report clocks

```
report clocks
     [-ideal] [-generated]
     [clock]... [-mode mode_name] [> file]
```

Generates a report on the clocks of the current design. Reports the clock period, rise, fall, domain, setup uncertainty, latency, clock ports or sources in the current design.

Use the `-generated` option to report generated clock information, and use the `-ideal` option to report an ideal clock - clock relationship.

Options and Arguments

clock	Specifies the name of the clock for which you want to generate the report.
file	Specifies the name of the file to which to write the report.
`-generated`	Adds generated clock information to the description, uncertainty, and the relationship table.
`-ideal`	Reports a clock description with the ideal clock - clock relationship.
`-mode` *mode_name*	Generates a report by mode on the clocks of the current design.

Example

The following example generates a clock report with generated clock information added to the table.

```
legacy_genus:/> report clocks -generated
============================================================
  Generated by:           Genus Synthesis Solution Version
  Generated on:           Date
  Module:                 test
  Technology library:     tutorial 1.0
  Operating conditions:   typical_case (balanced_tree)
  Wireload mode:          enclose

============================================================
 Clock Description
 -----------------
 Clock                                                No of
 Name    Period  Rise    Fall     Domain  Pin/Port  Registers
-------------------------------------------------------------
 CLK1    4000.0   0.0   2000.0   domain_1   Clk             5
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
 CLK2    2000.0   0.0   1000.0   domain_1   C              0
 CLK3    3000.0   0.0   1500.0   domain_2   Clk1           5
 CLK4    6000.0   0.0   3000.0   domain_2   C1             0
 Clock Network Latency / Setup Uncertainty
 -----------------------------------------
        Network   Network   Source   Source      Setup         Setup
Clock   Latency   Latency  Latency  Latency  Uncertainity  Uncertainity
 Name    Rise      Fall      Rise     Fall       Rise          Fall
-----------------------------------------------------------------------
CLK1      140.0     140.0    150.0    150.0         100.0         100.0
CLK2      120.0     120.0    120.0    120.0         110.0         110.0
CLK3      100.0     100.0    100.0    100.0         100.0         100.0
CLK4        0.0       0.0      0.0      0.0           0.0           0.0
 Clock Relationship (with uncertainity & latency)
 ------------------------------------------------
  From    To     R->R     R->F     F->R     F->F
--------------------------------------------------
  CLK1   CLK1   3900.0   1900.0   1900.0   3900.0
  CLK1   CLK2   1840.0    840.0   1840.0    840.0
  CLK2   CLK1   1950.0   1950.0    950.0    950.0
  CLK2   CLK2   1890.0    890.0    890.0   1890.0
  CLK3   CLK3   2900.0   1400.0   1400.0   2900.0
  CLK3   CLK4   2800.0   2800.0   1300.0   1300.0
  CLK4   CLK3   3100.0   1600.0   3100.0   1600.0
  CLK4   CLK4   6000.0   3000.0   3000.0   6000.0
```

Related Information

Affected by this command: create_mode on page 453

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report congestion

```
report congestion [ >file]
```

Reports the percentage of gcells with overflow, the total overflow of the design as well as the maximum overflow and the associated gcell.

Options and Arguments

file	Specifies the name of the file to which to write the report.

Example

The following command shows the congestion summary for design `test`.

```
legacy_genus:/> report congestion
============================================================
  Generated by:           Genus Synthesis Solution Version
  Generated on:           Date
  Module:                 test
  Technology libraries:   lib1
                          lib2
  Operating conditions:   max
  Interconnect mode:      placement
  Area mode:              physical library
============================================================

GCELLS with H overflow: 99 (6.71%)
GCELLS with V overflow: 329 (22.29%)
Total number of GCELLS: 1476

Item                   Oflow/Avail   Gcell (Bounding-box location)
-------------------------------------------------------------------
Max. Overflow          -111/201      20,18 (480.0,432.0),(504.0,456.0)
Max. Overflow (H)      -52/118       31,29 (744.0,696.0),(768.0,720.0)
Max. Overflow (V)      -75/87        19,17 (456.0,408.0),(480.0,432.0)
Total Overflow         -7172
Total Overflow (H)     -1329
Total Overflow (V)     -5843
```

Related Information

Affected by these commands: syn_generic -physical

syn_opt -physical

syn_opt -spatial

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report datapath

```
report datapath [-full_path]
     [-no_header] [-no_area_statistics]
     [-mux] [-all] [-max_width string]
     [-print_instantiated] [-print_inferred]
     [-sort keys] [design] [> file]
```

Reports datapath operators that were inferred from the design. This command is available after elaboration. You must set the set the `hdl_track_filename_row_col` attribute to `true` to enable filename, column, and line number tracking in the datapath report; otherwise these headings will be hidden.

Options and Arguments

`-all`	Reports all datapath operators present in the design including muxes. **Note:** The mux operators are different from the MUX library cells that are picked by the mapper or are available in the technology library.
`design`	Specifies a particular design on which to report datapath operators. By default, Genus reports on the current design.
`file`	Specifies the name of the file to which to write the report.
`-full_path`	Reports the full UNIX path name of the filename, including the filename. By default, Genus only reports the design name. See Examples for more information.
`-max_width` *`string`*	Specifies the maximum width of individual column names. By default, the maximum width for a column is 20. If a name is more than 20, it will wrap to the next line. The valid column names are *Operator*, *Signedness*, *Inputs, Outputs*, *Cell Area*, *Line*, *Col*, and *Filename*.
`-mux`	Reports muxes present in the design. Muxes are not reported by default. Using the `-mux` option only displays the muxes in the design and suppresses the other datapath operators. To view both, use the `-all` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-no_area_statistics` Suppresses the table that only shows the total area and percentage information. The area and the percentage of the total area consumed by the datapath operators in the design are only available after issuing the `synthesize -to_mapped` command.

`-no_header` Suppresses the header information.

`-print_inferred` Reports only the inferred datapath components in the design.

`-print_instantiated` Reports only the instantiated datapath components in the design.

`-sort` *`keys`* Indicates how to sort the report. You can sort on the following keys:

- `architectures` sorts the architectures in alphabetical order
- `area` sorts by descending area
- `inputs` sorts based on the number*width (number of bits) of the input signals—components with higher number of bits are printed first
- `instances` sorts the instances in alphabetical order
- `outputs` sorts based on the number*width (number of bits) of the output signals—components with higher number of bits are printed first
- `operators` sorts by operator
- `slack` sorts by ascending slack
- `subdesigns` sorts the subdesigns in alphabetical order

By default, the report does not contain the slack numbers.

Note: You can sort on several keys.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example generates a report of the datapath components for the `rpdp1_basic` design.

```
legacy_genus:/> report datapath rpdp1_basic
============================================================
...
  Module:                 rpdp1_basic
  Technology library:     tutorial 1.0
...
============================================================
Instantiated datapath components

              Operator Signedness  Inputs Outputs CellArea Line Col  Filename
================================================================================
rpdp1_basic
 d1
  u2
  module:CW__CW_multadd__builtin_wA8_wB8_wC8_wZ16
   CW/CW_multadd/builtin
                 n/a   n/a        8x8x8x1 16       1239.75   38  52 impl_inf.v
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   mul_1_19
    very_fast/non_booth
                  *    signed     9x9     16        912.75
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   add_1_24
    very_fast     +    signed     16x9    16        322.50
================================================================================

Inferred components

              Operator Signedness  Inputs Outputs CellArea Line Col  Filename
================================================================================
rpdp1_basic
 mul_18_28
 module:mult_unsigned
  very_fast/non_booth
                  *    unsigned   16x8    16       1044.00   18  28 impl_inf.v
================================================================================

      Type       CellArea Percentage
-------------------------------------
datapath modules  2283.75      20.55
external muxes       0.00       0.00
others            8829.00      79.45
-------------------------------------
total            11112.75     100.00
```

- By default, when using the `report datapath` command on a mapped netlist containing datapath operators, you will get the area statistics of the design, as shown in the following example.

```
-----------------------------------
datapath modules 4938.00     100.00
mux modules         0.00       0.00
others              0.00       0.00
-----------------------------------
total            4938.00     100.00
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This information is useful in determining the percentage of the design that contains datapath operators. If you do not want to report this information, then use the `-no_area_statistics` option.

By default, the area report is suppressed for a netlist that contains only generic cells (no library cells).

- The following command provides a 30-character width to the *filename* column and provides a 0-character width to the *area* column.

```
report datapath -max_width {{filename 30} {area 0}}
```

- The following command generates a report sorted by area.

```
legacy_genus:/> report datapath -sort area
============================================================
...
============================================================


Inferred components

            Operator Signedness Inputs Outputs CellArea Line Col Filename
============================================================================
lt_leq
 add_14_26
 module:add_signed
  very_fast     +    signed     4x4    4          31.50   14  26 lt_leq.v
============================================================================
lt_leq
 lt_13_25
 module:lt_signed
  very_fast     <    signed     4x4    1          26.25   13  25 lt_leq.v
============================================================================
lt_leq
 lt_16_25
 module:lt_unsigned
  very_fast     <    unsigned   4x4    1          26.25   16  25 lt_leq.v
============================================================================
lt_leq
 le_17_26
 module:leq_unsigned
  very_fast    <=    unsigned   4x4    1          25.50   17  26 lt_leq.v
============================================================================

      Type       CellArea Percentage
-------------------------------------
datapath modules   109.50      75.26
external muxes       0.00       0.00
others              36.00      24.74
-------------------------------------
total              145.50     100.00
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command sorts the report first by slack, then by area.

```
legacy_genus:/> report datapath -sort {slack area}
============================================================
...
============================================================


Inferred components

          Operator Signedness Inputs Outputs CellArea Line Col Filename  Slack
================================================================================
====
lt_leq
 lt_13_25
 module:lt_signed
  very_fast     <    signed     4x4    1         26.25   13  25 lt_leq.v -145.2
==============================================================================
lt_leq
 lt_16_25
 module:lt_unsigned
  very_fast     <    unsigned   4x4    1         26.25   16  25 lt_leq.v -145.2
==============================================================================
lt_leq
 le_17_26
 module:leq_unsigned
  very_fast    <=    unsigned   4x4    1         25.50   17  26 lt_leq.v -53.3
==============================================================================
lt_leq
 add_14_26
 module:add_signed
  very_fast     +    signed     4x4    4         31.50   14  26 lt_leq.v 49.0
==============================================================================

      Type       CellArea Percentage
-------------------------------------
datapath modules   109.50      75.26
external muxes       0.00       0.00
others              36.00      24.74
-------------------------------------
total              145.50     100.00
```

Related Information

Affected by this attribute: [hdl_track_filename_row_col]

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report design_rules

```
report design_rules [design]...
     [-include_max_lut_check] [> file]
```

Reports any design rule violations that are present in the specified design objects.

Options and Arguments

design
: Specifies the design for which you want to generate a report.

  By default, a report is created for all designs currently loaded in memory.

file
: Specifies the name of the file to which to write the report.

-include_max_lut_check
: Reports the following additional information when frequency-based maximum capacitance calculation is enabled (using the `use_max_cap_lut` attribute) during design rule fixing and verification:

  - The instances that are operating at a frequency outside of the cell's `max_cap` lookup table range
  - The set and count of instantiated technology cells that did not have a `max_cap` lookup table

Examples

- The following example generates a report of the design rule violations for the specified design.

```
> report design_rules alu
============================================================
  Generated by:          Genus Synthesis Solution version
  Generated on:          Current date Current time
  Module:                alu
  Technology library:    tutorial 1.0
  Operating conditions:  typical_case (balanced_tree)
  Wireload mode:         enclosed
============================================================

Max_transition design rule: no violations.


Max_capacitance design rule: no constraints.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Max_fanout design rule: no constraints.
```

- The following example generates a report of the design rule violations when frequency-based maximum capacitance calculation is enabled during design rule verification.

```
> report design_rules alu  -include_max_lut_check
============================================================
  Generated by:          Genus Synthesis Solution version
....
============================================================
Cells with no max capacitance lut table(total=6)
Cell       Library    # of instances
-----------------------------------
AND2X2    mylib                 1
BUFX12    mylib                 1
BUFX16    mylib                 2
DFFHQX4   mylib             20002
INVX16    mylib                 1
NAND2XL   mylib             10001

Instances with frequency constraint outside cell's max capacitance lut's rang
e
(total=294)
Instance   Cell             Freq(Mhz)    Slowest(Mhz)    Fastest(Mhz)
-------------------------------------------------------------------------
mm/fopt    BUFX20           10000.000           0.100        1000.000
mm/fopt1   BUFX20           10000.000           0.100        1000.000
mm/fopt10  BUFX20           10000.000           0.100        1000.000
mm/fopt100 BUFX20           10000.000           0.100        1000.000
mm/fopt101 BUFX20           10000.000           0.100        1000.000
mm/fopt102 BUFX20           10000.000           0.100        1000.000
```

Related Information

Affected by these commands:	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	ignore_library_max_fanout
	max_capacitance
	max_fanout
	max_transition
	use_max_cap_lut

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_chains

```
report dft_chains [design] [-chain actual_scan_chain]...
     [-dft_configuration_mode dft_config_mode_name]
     [-opcg_side_scan]
     [-summary] [> file]
```

Reports for every chain in the design the following elements:

- The scan data input port and its hookup pin
- The scan data output port and its hookup pin
- The shift enable port and its hookup pin
- The DFT clock domain and the DFT clock domain edge of the chain (for muxed scan only)
- The length of the chain
- The elements in the chain

  In case of the muxed scan style, the report also lists for each element the test clock and test clock edge determined by the `check_dft_rules` command for that element.
- Any user-specified segments with their elements

  In case of the muxed scan style, this includes data lockup elements, and the location of the data lockup element in the chain.
- The bit position and length of any user-specified segment in the chain
- The head and tail test clock and test clock edge for any abstract segment in the chain

Options and Arguments

`-chain` *`actual_scan_chain...`*

Reports only the listed scan chains

`design`

Specifies the design for which you want to generate a report.

Note: If you have multiple top designs, you must either specify the design name, or change to the directory in the design hierarchy that contains the design for which you want the report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dft_configuration_mode` *`dft_configuration_mode_name`*

Reports the scan chains related to the specified scan mode.

`file` Specifies the name of the file to which to write the report.

`-opcg_side_scan` Reports only the side scan chains that contain the domain macro segments.

`-summary` Condenses the scan chain report to include only chain name, scan-data pins, shift enable, clock domain and length information.

Examples

- The following example shows one scan chain, with three user-defined segments.

```
legacy_genus:/designs/test> report dft_chains
Reporting 1 scan chain

Chain 1: DFT_chain_1
  scan_in:      in[0]
  scan_out:     out[1] (shared output)
  shift_enable: SE (active high)
clock_domain: clk (edge: mixed)
  length: 8
    START segment segHead (type: floating)
      # @ bit 1, length: 2
     bit 1     out_reg_4 <clk/fall>
     bit 2     out_reg_5 <clk/fall>
    END segment segHead
    bit 3       out_reg_6 <clk/fall>
    bit 4       out_reg_7 <clk/fall>
    START segment segBody (type: fixed)
      # @ bit 5, length: 2
      bit 5     out_reg_1 <clk/rise>
      bit 6     out_reg_3 <clk/rise>
    END segment segBody
    bit 7       out_reg_2
    START segment segTail (type: floating)
      # @ bit 8, length: 1
      bit 8     out_reg_0 <clk/rise>
    END segment segTail
------------------------
```

- The following example shows the summary report for the previous example.

```
legacy_genus:/designs/test> report dft_chains -summary
Reporting 1 scan chain (muxed_scan)

Chain 1: DFT_chain_1
  scan_in:      in[0]
  scan_out:     out[1] (shared output)
  shift_enable: SE (active high)
  clock_domain: clk (edge: mixed)
  length: 8
------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following examples show the scan chains of a design before and after DFT compression.

  - **Before compression**

```
Reporting 2 scan chains (muxed_scan)

Chain 1: AutoChain_1
  scan_in:      DFT_sdi_1
  scan_out:     DFT_sdo_1
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
    bit 1       out_reg[1]  <clk (rise)>
    bit 2       out_reg[2]  <clk (rise)>
    bit 3       out_reg[3]  <clk (rise)>
    bit 4       out_reg[4]  <clk (rise)>
    bit 5       out_reg[5]  <clk (rise)>
    bit 6       out_reg[6]  <clk (rise)>
    bit 7       out_reg[7]  <clk (rise)>
    bit 8       out_reg[8]  <clk (rise)>
    bit 9       out_reg[9]  <clk (rise)>
    bit 10      out_reg[10]  <clk (rise)>
------------------------
Chain 2: AutoChain_2
  scan_in:      DFT_sdi_2
  scan_out:     DFT_sdo_2
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
    bit 1       out_reg[11]  <clk (rise)>
    bit 2       out_reg[12]  <clk (rise)>
    bit 3       out_reg[13]  <clk (rise)>
    bit 4       out_reg[14]  <clk (rise)>
    bit 5       out_reg[15]  <clk (rise)>
    bit 6       out_reg[16]  <clk (rise)>
    bit 7       out_reg[17]  <clk (rise)>
    bit 8       out_reg[18]  <clk (rise)>
    bit 9       out_reg[19]  <clk (rise)>
    bit 10      out_reg[20]  <clk (rise)>
------------------------
```

  - **After compression**

    The report shows in addition

    - How the original scan chains have been divided in several internal chains.
    - For each internal chain the `START` and `END` (separate scan data input and output) are given together with the length of the internal channel.
    - An additional scan chain, `mask_chain` (if the user opted to add making logic)

      The mask chain is comprised of abstract segments only.

      The shift-enable signal of the mask chain is reported as `NONE` because it is a *connected* shift enable.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Reporting 3 scan chains (muxed_scan)

Chain 1: AutoChain_1 (compressed)
  scan_in:      DFT_sdi_1
  scan_out:     DFT_sdo_1
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
      <START compressed internal chain AutoChain_1_0 (sdi: g121/SWBOX_SI[0])>
    bit 1       out_reg[1]  <clk (rise)>
    bit 2       out_reg[2]  <clk (rise)>
      <END   compressed internal chain AutoChain_1_0 (sdo: g121/SWBOX_SO[0])
(length: 2)>
      <START compressed internal chain AutoChain_1_1 (sdi: g121/SWBOX_SI[1])>
    bit 3       out_reg[3]  <clk (rise)>
    bit 4       out_reg[4]  <clk (rise)>
      <END   compressed internal chain AutoChain_1_1 (sdo: g121/
SWBOX_SO[1]) (length: 2)>
      <START compressed internal chain AutoChain_1_2 (sdi: g121/SWBOX_SI[2])>
    bit 5       out_reg[5]  <clk (rise)>
    bit 6       out_reg[6]  <clk (rise)>
      <END   compressed internal chain AutoChain_1_2 (sdo: g121/SWBOX_SO[2])
(length: 2)>
      <START compressed internal chain AutoChain_1_3 (sdi: g121/SWBOX_SI[3])>
    bit 7       out_reg[7]  <clk (rise)>
    bit 8       out_reg[8]  <clk (rise)>
      <END   compressed internal chain AutoChain_1_3 (sdo: g121/SWBOX_SO[3])
(length: 2)>
      <START compressed internal chain AutoChain_1_4 (sdi: g121/SWBOX_SI[4])>
    bit 9       out_reg[9]  <clk (rise)>
    bit 10      out_reg[10]  <clk (rise)>
      <END   compressed internal chain AutoChain_1_4 (sdo: g121/SWBOX_SO[4])
(length: 2)>
------------------------
Chain 2: AutoChain_2 (compressed)
  scan_in:      DFT_sdi_2
  scan_out:     DFT_sdo_2
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
      <START compressed internal chain AutoChain_2_5 (sdi: g121/SWBOX_SI[5])>
    bit 1       out_reg[11]  <clk (rise)>
    bit 2       out_reg[12]  <clk (rise)>
      <END   compressed internal chain AutoChain_2_5 (sdo: g121/SWBOX_SO[5])
(length: 2)>
...
...
      <START compressed internal chain AutoChain_2_9 (sdi: g121/SWBOX_SI[9])>
    bit 9       out_reg[19]  <clk (rise)>
    bit 10      out_reg[20]  <clk (rise)>
      <END   compressed internal chain AutoChain_2_9 (sdo: g121/SWBOX_SO[9])
(length: 2)>
------------------------
Warning - could not find shift_enable signal for chain /designs/top/dft/
report/actual_scan_chains/mask_chain
Chain 3: mask_chain
  scan_in:      msi
  scan_out:     mso
  shift_enable: NONE
  clock_domain: mck (edge: rise)
  length: 10
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
    START segment DFT_segment_1 (type: abstract)
      # @ bit 1, length: 4
      pin       g121/msi[0] <mck (rise)>
      pin       g121/mso[0] <mck (rise)>
    END segment DFT_segment_1
    START segment DFT_segment_3 (type: abstract)
      # @ bit 5, length: 3
      pin       g121/msi[2] <mck (rise)>
      pin       g121/mso[2] <mck (rise)>
    END segment DFT_segment_3
    START segment DFT_segment_2 (type: abstract)
      # @ bit 8, length: 3
      pin       g121/msi[1] <mck (rise)>
      pin       g121/mso[1] <mck (rise)>
    END segment DFT_segment_2
------------------------
```

- The following command reports two side scan chains, each with one `OPCG` segment.

```
legacy_genus:/> report dft_chains -opcg_side_scan

Reporting 2 side scan chains
Side-scan chain 1:
  scan_in:      io[4] (hookup: p/p4/Y)
  clock_domain: opcg_load_clk
  length: 7
    START segment OPCG_DOMAIN_SEG_1 (type: abstract)
      #@bit 1, length 7
      pin        DFT_OD2/PGMSI
      pin        DFT_OD2/PGMSO
    END segment OPCG_DOMAIN_SEG_1
------------------------
Side-scan chain 2:
  scan_in:      DFT_sdi_1
  clock_domain: opcg_load_clk
  length: 6
    START segment OPCG_DOMAIN_SEG_0 (type: abstract)
      #@bit 1, length 6
      pin        DFT_OD1/PGMSI
      pin        DFT_OD1/PGMSO
    END segment OPCG_DOMAIN_SEG_0
------------------------
```

Related Information

Affected by this command:	connect_scan_chains on page 906
	compress_scan_chains on page 882
Related attributes:	Actual Scan Chain attributes
	Actual Scan Segment attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_clock_domain_info

```
report dft_clock_domain_info
     [-endpoint_limit integer]
     [-system_clock_violations]
     [-report_internal] [-verbose]
     [design] [> file]
```

Reports the DFT clock domain information.

This command should be run after `connect_scan_chains` command.

Options and Arguments

design	Specifies the design for which you want to report the DFT clock domain information.
`-endpoint_limit` *integer*	Prevents reporting endpoints if the number is less than the specified limit.
`-report_internal`	Reports the number of endpoints that receive data from the same clock domain
`-system_clock_violations`	Reports only system clock violations.
`-verbose`	Lists all endpoints in the report.

Examples

- Sample output:

```
report_opcg_clock_domain_info

Test clock domain report:
Clock domain 1: opcg_load_clk
Test clock 1: opcg_load_clk (root pin: opcg_load_clk)
       Total endpoints: 0
       System mode clock control: 0 clock pins.
              Total 0 endpoints exempted from system clock control check.
       Domain crossing capture report: total 0 endpoints and 0 domains.
       Domain crossing lauching report: total 0 endpoints and 0 domains.

Clock domain 2: scanclk
Test clock 1: scanclk (root pin: scanclk)
       Total endpoints: 82
       System mode clock control: 1 clock pins.
              Total 0 endpoints exempted from system clock control check.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
                82 endpoints are controlled by system clock scanclk.
        Domain crossing capture report: total 0 endpoints and 0 domains.
        Domain crossing lauching report: total 0 endpoints and 0 domains.
Test clock 2: DFT_test_clock_OD1 (root pin: scanclk)
        Total endpoints: 5
        System mode clock control: 1 clock pins.
                Total 0 endpoints exempted from system clock control check.
                5 endpoints are controlled by system clock
                DFT_OD1/i_opcg_clkout_mux/opcg_clk_mux_OD1_p2_d7_b1_opcg_
                clkout_mux/mux_1_19_g1/Y.
        Domain crossing capture report: total 5 endpoints and 1 domains.
        5 endpoints receive data from domain 'UNKNOWN'.
        Domain crossing lauching report: total 0 endpoints and 0 domains.
```

Related Information

Related command: insert_dft opcg on page 1063

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_core_wrapper

```
report dft_core_wrapper
     [-wrapped | -unwrapped]
     [-location {pin|port|subport}...]
     [-report_flops] [-summary]
     [-max_print_objects integer]
     [design|instance] [>file]
```

Reports the wrapper cells inserted in the design.

Options and Arguments

`{design|instance}`	Specifies the design or instance for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
`file`	Specifies the name of the file to which to write the report.
`-location {pin|port|subport}`	Limits the reported wrapper cells to the specified pin(s), port(s) or subport(s).
`-max_print_objects integer`	Specifies the maximum number of wrapper segments or flops to be reported on any pin, port or subport. *Default:* `5`
`-report_flops`	Reports the flops associated with the wrapper segments.
`-summary`	Reports the summary only.
`-unwrapped`	Lists only those ports or pins that do not have a wrapper cell associated with them.
`-wrapped`	Lists only those ports or pins that have an associated wrapper cell.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following examples illustrate the different report options on the same design.

```
legacy_genus:/> report dft_core_wrapper test
Reporting 13 ports/pins
 Port/Pin                        type                 usage                  wrapper_objects
 ========                        ====                 =====                  ===============
 WEXT                            wrapper_control        -                      -
 WIG                             wrapper_control        -                      -
 WINT                            wrapper_control        -                      -
 WOG                             wrapper_control        -                      -
 WSEN_in                         wrapper_control        -                      -
 clk                             wrapper_control        -                      -
 WSEN_out                        shift_enable           -                      -
 se                              shift_enable           -                      -
 in1                             shared               input                  wrap_1
                                                                             wrap_4
 in1                             shared               output                 wrap_3
 in2                             shared               input                  wrap_1
                                                                             wrap_4
 in2                             shared               output                 wrap_3
 in3                             shared               input                  wrap_2
 out1                            shared               input                  wrap_1
                                                                             wrap_4
 out2                            shared               input                  wrap_1
                                                                             wrap_4

Wrapper insertion summary report
================================
Number of Dedicated wrapper cells :0     input bounding :0    output bounding :0
Number of Shared wrapper cells    :4     input bounding :3    output bounding :1
--------------------------------------------------------------------------------
Total no. of wrapper cells        :4     input bounding :3    output bounding :1
```

```
legacy_genus:/> report dft_core_wrapper test -wrapped
Reporting 13 ports/pins
 Port/Pin                        type                 usage                  wrapper_objects
 ========                        ====                 =====                  ===============
 in1                             shared               input                  wrap_1
                                                                             wrap_4
 in1                             shared               output                 wrap_3
 in2                             shared               input                  wrap_1
                                                                             wrap_4
 in2                             shared               output                 wrap_3
 in3                             shared               input                  wrap_2
 out1                            shared               input                  wrap_1
                                                                             wrap_4
 out2                            shared               input                  wrap_1
                                                                             wrap_4

Wrapper insertion summary report
================================
Number of Dedicated wrapper cells :0     input bounding :0    output bounding :0
Number of Shared wrapper cells    :4     input bounding :3    output bounding :1
--------------------------------------------------------------------------------
Total no. of wrapper cells        :4     input bounding :3    output bounding :1
```

```
legacy_genus:/> report dft_core_wrapper -report_flops
Reporting 13 ports/pins
 Port/Pin        type              Usage                 wrapper_objects                wrapper_flops
-------------------------------------------------------------------------------------------------------

 WEXT         wrapper_control        -                     -
 WIG          wrapper_control        -                     -
 WINT         wrapper_control        -                     -
 WOG          wrapper_control        -                     -
 WSEN_in      wrapper_control        -                     -
 clk          wrapper_control        -                     -
 WSEN_out     shift_enable           -                     -
 se           shift_enable           -                     -
 in1          shared               input                 wrap_1                        tmp5_reg
 in1          shared               input                 wrap_4                        tmp4_reg
 in1          shared               output                wrap_3                        tmp3_reg
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
 in2          shared               input                 wrap_1                        tmp5_reg
 in2          shared               input                 wrap_4                        tmp4_reg
 in2          shared               output                wrap_3                        tmp3_reg
 in3          shared               input                 wrap_2                        tmp2_reg
 out1         shared               input                 wrap_1                        tmp5_reg
 out1         shared               input                 wrap_4                        tmp4_reg
 out2         shared               input                 wrap_1                        tmp5_reg
 out2         shared               input                 wrap_4                        tmp4_reg

Wrapper insertion summary report
================================
Number of Dedicated wrapper cells :0     input bounding :0    output bounding :0
Number of Shared wrapper cells    :4     input bounding :3    output bounding :1
-------------------------------------------------------------------------------
Total no. of wrapper cells        :4     input bounding :3    output bounding :1
```

```
legacy_genus:/> report dft_core_wrapper test -summary
Wrapper insertion summary report
================================
Number of Dedicated wrapper cells :0     input bounding :0    output bounding :0
Number of Shared wrapper cells    :4     input bounding :3    output bounding :1
-------------------------------------------------------------------------------
Total no. of wrapper cells        :4     input bounding :3    output bounding :1
```

Related Information

Affected by this command: insert_dft wrapper_cell on page 1094

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_registers

```
report dft_registers [-pass_tdrc] [-fail_tdrc]
     [-lockup] [-latch] [-dont_scan] [-misc]
     [-shift_reg] [-multi-bit]
     [-test_clock test_clock [-test_clock_edge test_clock_edge]]
     [design] [> file]
```

Reports the DFT status of all flip-flop instances in the design. Use this command after running `check_dft_rules`. More specifically, the command reports

- Registers which pass the DFT rule checker

  For each flip-flop, it reports the hierarchical instance name along with their DFT test clock domain and active edge.

- Registers which fail the DFT rule checker

  For each flip-flop, it reports the hierarchical instance name along with the violation type (clock, or asynchronous set/reset) and the violation Id number.

  For an abstract segment that fails the DFT rule checker, it reports the name of the abstract segment, and the number of flip-flops in the segment.

- Registers that are preserved or marked `dont-scan`

  Note: Mapped flip-flops can be listed in this category if

  - The flip-flop is marked preserved and it is mapped to a non-scan flop

    However, if a flip-flop is marked preserved and is already mapped to scan for DFT purposes, it will be listed as either passing or failing the DFT rule checker—based on the DFT rule checker analysis—and it will not be listed in this category.

  - The flip-flop is mapped to a scan flop for non-DFT purposes

- Registers that are marked Abstract Segment dont-scan.
- Registers that are part of shift register segments
- Registers that are identified as lockup elements
- Registers that are level-sensitive elements
- Registers that are implemented as multibit
- Registers not part of any of the other categories

Without any options specified, the command reports on all categories of registers.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

design	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
-dont_scan	Reports all edge-triggered registers that are not to be mapped to scan flops, or connected into the top-level chains.
-fail_tdrc	Reports all edge-triggered registers that fail the DFT rule checks.
file	Specifies the name of the file to which to write the report.
-latch	Reports all registers of type latch in the design. **Note:** Latch instances which are instantiated as lockup elements in a preserved segment are reported with the -lockup option.
-lockup	Reports data lockup elements of type latch or flop that are either instantiated in a preserved segment, or added to the design during scan chain configuration.
-misc	Reports all registers that are not reported through any of the other options, such as clock-gating registers.
-multi_bit	Reports all registers that are implemented as multibit elements.
-pass_tdrc	Reports all edge-triggered registers that pass the DFT rule checks.
-shift_reg	Reports all registers that are part of shift register segments.
-test_clock *test_clock*	Returns the number of registers controlled by the specified test clock.
-test_clock_edge *test_clock_edge*	Returns the number of registers controlled by the specified test clock edge. **Note:** This option must be specified with the -test_clock option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example shows that 1 flip-flop passed the DFT rule checks, while 4 flip-flops failed the tests.

```
legacy_genus:/> report dft_registers

Reporting registers that pass DFT rules
  Iset_reg      PASS; Test clock: clk/rise
Reporting registers that fail DFT rules
  out_reg_0     FAIL; violations: clock #(0 ) async set #(1 )
  out_reg_1     FAIL; violations: clock #(0 ) async set #(1 )
  out_reg_2     FAIL; violations: clock #(0 ) async set #(1 )
  out_reg_3     FAIL; violations: clock #(0 ) async set #(1 )
Reporting registers that are preserved or marked dont-scan
Reporting registers that are marked Abstract Segment Dont Scan
Reporting registers that are part of shift register segments
Reporting registers that are identified as lockup elements
Reporting registers that are level-sensitive elements
Reporting misc. non-scan registers
Summary:
Total registers that pass DFT rules: 1
Total registers that fail DFT rules: 4
Total registers that are marked preserved or dont-scan: 0
Total registers that are marked Abstract Segment dont-scan: 0
Total registers that are part of shift register segments: 0
Total registers that are lockup elements: 0
Total registers that are level-sensitive: 0
Total registers that are misc. non-scan: 0
```

- The following report was generated after the scan configuration engine was run. In this case, four lockup elements were inserted.

```
legacy_genus:/> report dft_registers

Reporting registers that pass DFT rules
  out1_reg_0    PASS; Test clock: test_clk1/rise
  ...
  out1_reg_4    PASS; Test clock: test_clk1/fall
   ...
  out2_reg_0    PASS; Test clock: test_clk2/rise
   ...
  out2_reg_4    PASS; Test clock: test_clk2/fall
  ...
  out3_reg_0    PASS; Test clock: test_clk3/rise
  ...
  out3_reg_4    PASS; Test clock: test_clk3/fall
  ...
Reporting registers that fail DFT rules
Reporting registers that are preserved or marked dont-scan
Reporting registers that are marked Abstract Segment Dont Scan
Reporting registers that are part of shift register segments
Reporting registers that are identified as lockup elements
  DFT_lockup_g1
  DFT_lockup_g348
  DFT_lockup_g349
  DFT_lockup_g350
Reporting registers that are level-sensitive elements
Reporting misc. non-scan registers
Summary:
Total registers that pass DFT rules: 27
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Total registers that fail DFT rules: 0
Total registers that are marked preserved or dont-scan: 0
Total registers that are marked Abstract Segment dont-scan: 0
Total registers that are part of shift register segments: 0
Total registers that are lockup elements: 4
Total registers that are level-sensitive: 0
Total registers that are misc. non-scan: 0
```

Related Information

Affected by this command:	check_dft_rules on page 870
Related attributes:	dft_dont_scan
	dft_scan_style
	dft_status
	dft_violation
	preserve (instance)
	preserve (subdesign)
	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_setup

`report dft_setup [design] [> file]`

Reports DFT setup information for the design including both user-defined and tool-inferred information.

You must load and elaborate a design before you can use this command. The contents of this report further depends on the stage of the design that you are at.

Use this command at the end of the design process to document the DFT setup and resulting configuration of the design.

Options and Arguments

design	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
file	Specifies the name of the file to which to write the report.

Examples

- The following example shows the report after you have elaborated the design. Because you have not specified any setup information yet, the information shown is based on the default settings for DFT.

```
legacy_genus:/> report dft_setup

Design Name
===========
   top

Scan Style
==========
    muxed_scan
DFT rule check status is not available. Need to (re)run check_dft_rules

Global Constraints
==================
    Minimum number of scan chains: no_value
    Maximum length of scan chains: no_value
    Lock-up element type: level_sensitive
    Mix clock edges in scan chain: false
    Prefix for unnamed scan objects: DFT_

Test signal objects
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
===================

Test clock objects
===================
Reporting all test clocks as TDRC status is not available

DFT controllable objects
=========================

DFT don't scan objects
=======================

DFT abstract don't scan objects
================================

DFT scan segment constraints
============================

DFT scan chain constraints
==========================

DFT actual scan chains
======================
```

- The following example shows the report after the scan configuration engine was run.

```
legacy_genus:/designs/test> report dft_setup

Design Name
===========
   top

Scan Style
==========
    muxed_scan
Design has a valid DFT rule check status

Global Constraints
==================
    Minimum no of Scan chains: no_value
    Maximum length of scan chains: no_value
    Lock-up element type: level_sensitive
    Mix clock edges in scan chain: true
    Prefix to name user defined scan chains: DFT_

Test signal objects
===================
    shift_enable:
          object name: SE
          pin name: SE
          hookup_pin: SE
          hookup_polarity: non_inverted
          active: high
          ideal: true
          user defined: true


Test clock objects
===================
    test_clock:
          object name: clk
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
          user defined: false
          source: clk
          root source: clk
          root source polarity: non_inverting
          hookup_pin: clk
          period: 50000.0


DFT controllable objects
========================

DFT don't scan objects
======================

DFT abstract don't scan objects
===============================

DFT scan segment constraints
============================
    Segment:
          object name: segHead
          scan-in:
          scan-out:
          shift-enable: internal
          connected_shift_enable: false
          length: 2
          type: floating

    Segment:
         object name: segTail
         ...

    Segment:
          object name: segBody
          scan-in:
          scan-out:
          shift-enable: internal
          connected_shift_enable: false
          length: 2
          type: fixed


DFT scan chain constraints
==========================
    User Chain:
          object name: DFT_chain_1
          scan-in: in[0]
          scan-in hookup_pin: in[0]
          scan-out: out[1]
          scan-out hookup_pin: out[1]
          shared out: true
          shift_enable object name:
          max length: no_value
          head segment: segHead
          tail segment: segTail
          complete: false


DFT actual scan chains
======================
    Actual Chain:
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
object name: DFT_chain_1
scan-in: in[0]
scan-in hookup_pin: in[0]
scan-out: out[1]
scan-out hookup_pin: out[1]
shared out: true
shift_enable: SE
length: 8
segment objects: segHead segBody segTail
analyzed: false
test_clock domain: clk
test_clock edge: any
```

Related Information

Recommended Flow in *Genus Design for Test Guide for Legacy UI*

Affected by these constraints:	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995
Affected by this command:	check_dft_rules on page 870
	connect_scan_chains on page 906
Related attributes:	dft_controllable
	dft_dont_scan
	dft_lockup_element_type
	dft_max_length_of_scan_chains
	dft_min_number_of_scan_chains
	dft_mix_clock_edges_in_scan_chains
	dft_prefix
	dft_scan_style
	(instance) preserve
	(subdesign) preserve
Related attributes:	Actual Scan Chain Attributes
	Actual Scan Segment Attributes
	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Scan Chain Attributes

Test Clock Attributes

Test Signal Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_violations

```
report dft_violations [-async] [-clock]
     [-abstract] [-shiftreg] [-tristate]
     [-race] [-xsource] [-xsource_by_instance]
     [design] [> file]
```

Reports for each violation the object name, the type of violation, the description of the violation, how to find the root pin or port of the problem, the source file and the line number where to find the problem, the number of registers affected, and the instance names of the affected registers. The report is sorted by violation type. Run the DFT rule checker (`check_dft_rules` command) to get meaningful information. This command extracts the list of all violations with the corresponding violation points as the `check_dft_rules` command is not verbose by default.

Without any options specified, the command reports on all types of violations.

Options and Arguments

`-abstract`	Reports all abstract segment test mode violations.
`-async`	Reports all async set and async reset violations.
`-clock`	Reports all clock violations.
`design`	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
`file`	Specifies the name of the file to which to write the report.
`-race`	Specifies to report potential race condition violations.
`-shiftreg`	Reports all shift register segment violations.
`-tristate`	Reports tristate design rules checking violations.
`-xsource`	Reports X-Source violations.
`-xsource_by_instance`	Reports X-Source violations by instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example shows that the design has four violations, but only detailed information on the clock violations is requested.

```
legacy_genus:/> report  dft_violations -clock
Total 4 violations (muxed_scan)

Clock Rule Violations
=====================
Reporting 2 clock violations

Violation #0:
  Object name: vid_0_clock
  Type: clock violation
  Description: [CLOCK-05] internal or gated clock signal
  Source: g1/z (test10.v:12)
  Number of registers affected: 4
  Affected registers:
    out1_reg[0]
    out1_reg[1]
    out1_reg[2]
    out1_reg[3]

Violation #1:
  Object name: vid_1_clock
  Type: clock violation
  Description: [CLOCK-06] clock signal driven by a sequential element
  Source: divClk_reg/q (test10.v:14)
  Number of registers affected: 4
  Affected registers:
    out2_reg[0]
    out2_reg[1]
    out2_reg[2]
    out2_reg[3]

Violation Rule Summary Report
=============================
[CLOCK-05] internal or gated clock signal : 1
[CLOCK-06] clock signal driven by a sequential element : 1
```

- The following example reports one tristate net contention violation.

```
legacy_genus:/> report dft_violations
Total 1 violation (muxed_scan)

Tristate Net Violations
=====================
Reporting 1 tristate net contention violations

Violation #0:
 Object name: vid_0_tristate_net
 Type: tristate net violation
 Description: [TRISTATE_NET-01] tristate net potentially driven by conflicting
values

  Tristate net affected: tbus
  Tristate net load pin affected: tbus
  Tristate drivers causing contention: b2/Y b1/Y b3/Y b4/Y b6/Y b8/Y b7/Y b5/Y
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
...
Violation Rule Summary Report
=============================
[TRISTATE_NET-01] tristate net potentially driven by conflicting values : 1
```

Related Information

Reporting the DFT Violations in *Genus Design for Test Guide for Legacy UI*

Affected by this command:	check_dft_rules on page 870
	fix_dft_violations on page 1010
Related attributes:	Violations Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report disabled_transparent_latches

```
report disabled_transparent_latches [> file]
```

Reports the disabled transparent latches and the `enable` to `Q` arcs that are not yet disabled. Transparent latches are latches with `enable` signal held constant at the active state. Without enabling transparent latches, paths through them cannot be traced.

Options and Arguments

file	Specifies the name of the file to which to write the report.

Related Information

Affected by this command:	enable_transparent_latches on page 96

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report gates

```
report gates [-library_domain library_domain_list]
     [-power] [-leakage_power] [-net_power] [-yield]
     [-instance_hier instance]
     [design] [> file]
```

Reports the technology library cells that were implemented (and identifies their originating libraries), the area of the cell instances, and the break up of the instances into timing models, sequential cells, integrated clock-gating cells, inverters, buffers, and logic gate cells. Optionally power information can be reported.

Note: Timing models can refer to memory cells, IPs, and so on.

Options and Arguments

design	Specifies the block for which you want to generate a report. You can also `cd` into the particular design directory and generate the report.
file	Specifies the name of the file to which to write the report.
`-instance_hier` *instance*	Restricts the reported information to the specified hierarchical instance.
`-library_domain` *library_domain_list*	Restricts the reported information to the specified library domains. **Note:** This option only applies when using CPF.
`-leakage_power`	Adds leakage power information only.
`-net_power`	Adds net power information only.
`-power`	Adds leakage power and internal power information.
`-yield`	Reports the yield cost and yield percentage values for each library cell.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example reports information such as the type of cells that were used, number of instances, area, and originating library of the current design.

```
legacy_genus:/> report gates
============================================================
...
  Module:                 alu
  Technology library:     tutorial 1.0
  Operating conditions:   typical_case (balanced_tree)
  Wireload mode:          enclosed
============================================================

  Gate  Instances   Area     Library
---------------------------------------
flopdrs         9   72.000    tutorial
inv1           35  105.000    tutorial
nand2         112  112.000    tutorial
nor2           37   55.500    tutorial
xor2           17   34.000    tutorial
---------------------------------------
total         210  378.500

   Type    Instances   Area   Area %
-------------------------------------
sequential         9   72.000   19.0
inverter          35  105.000   27.7
logic            166  201.500   53.2
-------------------------------------
total            210  378.500  100.0
```

- The following example shows the gate report for a multibit design. In this case, the tool uses the `(++)` annotation to mark the multibit cells. The rest of the report does not change.

```
   Gate          Instances  Area      Library
----------------------------------------------
DUAL_DFFQD (++)          2  0.000    tutorial
INVX1                    1  3.395    slow
NAND2X1                  1  5.092    slow
SDFFHQXL                 2  0.000    slow
----------------------------------------------
total                    6  8.487

(++) : Multibit gate, use report multibit_inferencing command for detailed
report
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows the additional power information in the report.
  - The first table lists the leakage and internal power of all instances per used library cell.
  - The second table shows the number and percentage of instances coming from each library, the leakage and internal power consumed by these instances, as well as the percentage of power consumed by these instances.
  - The third table shows the leakage and internal power of all sequential cells, inverters, and combinational cells, as well as the percentage of power that each type consumes.

```
legacy_genus:/> report gates -power
============================================================
...
============================================================

                                   Leakage     Internal
    Gate    Instances     Area     Power (nW)  Power (nW)   Library
-----------------------------------------------------------------------
ACHCONX2TH          1      28.856      59.948     372.198    slow_hvt
ADDFHX1             1      39.040     158.364     276.327    slow
...
XOR2XL             10     118.820     339.664     471.216    slow
XOR2XLTH          449    5335.018    6259.161   22146.660    slow_hvt
XOR3XL              1      28.856      83.655      80.446    slow
-----------------------------------------------------------------------
total           11980  109000.238  139743.990  505460.121

                                 Leakage  Leakage   Internal  Internal
 Library Instances Instances % Power (nW) Power %  Power (nW)  Power %
-------------------------------------------------------------------------
slow          1919        16.0  74607.161    53.4  168629.031     33.4
slow_hvt     10061        84.0  65136.830    46.6  336831.090     66.6

                                       Leakage  Leakage  Internal  Internal
  Type    Instances    Area    Area % Power (nW) Power %  Power (nW)  Power %
-----------------------------------------------------------------------------
sequential       212   7864.054    7.2  17285.068   12.4  112374.696     22.2
inverter        1684   5961.269    5.5   5752.468    4.1   22492.255      4.4
buffer            37    364.941    0.3   1877.474    1.3    3920.823      0.8
logic          10047  94809.9748  87.0 114828.979   82.2  366672.348     72.5
-----------------------------------------------------------------------------
total          11980 109000.2382  100.0 139743.990 100.0  505460.121    100.0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example reports the yield cost and yield percentage values for each library cell.

```
Gate  Instances   Area       Cost       Yield    Library
-------------------------------------------------------------
flopdrs        33  264.000  3.39278e-06  99.9997    tutorial
inv1          103   51.500   1.5022e-06  99.9998    tutorial
nand2         315  315.000  1.08311e-05  99.9989    tutorial
nor2           19   28.500  6.79989e-07  99.9999    tutorial
-------------------------------------------------------------
total         470  659.000  1.64061e-05  99.9984

   Type    Instances   Area  Area %
-------------------------------------
sequential        33 264.000   40.1
inverter         103  51.500    7.8
logic            334 343.500   52.1
-------------------------------------
total            470 659.000  100.0
```

Related Information

Affected by these commands: syn_generic on page 526
syn_map on page 529
syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report hierarchy

```
report hierarchy
     [-subdesign subdesign] [design] [> file]
```

Generates a report based on the design hierarchy starting from the top level module down to the leaf module. The report will take the following format:

```
level instance ( module ) <status>
status  : preserve_<value> -- indicating preserve hierachy or inherited_preserve
value
        : blackbox --  indicating  unresolved instance
```

Note: The `hdl_track_filename_row_col` attribute needs to be set to `true` before elaboration in order to successfully use this command.

Options and Arguments

design	Specifies a specific design to report when there are multiple designs.
file	Specifies the name of the file to which to write the report.
-subdesign *subdesign*	Specifies the subdesign for which a hierarchical report is required.

Example

- The following example reports the hierarchy of design `m1`.

```
==============================================================
Hierarchy Report Format :

level instance ( module ) <status>

status :    preserve_<value> -- indicating preserve hierachy or
inherited_preserve value
       :    blackbox --  indicating  unresolved instance
=================================================================

 0 m1
  1 m2 ( m2 )
     2 m3 ( m3 )
        3 m3_0 ( m3_0 )
           4 m3_0_0 ( m3_0_0 )
        3 m4 ( m4 )
           4 m5 ( m5 )
              5 m5_bbox ( m5_bbox ) blackbox
           4 m6 ( m6 )
     2 m2myclk ( m2myclk )
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by this attribute: hdl_track_filename_row_col

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report instance

```
report instance [-timing] [-power] [-detail]
     instance...  [> file]
```

Generates a report on the instances of the current design. By default, the report gives timing related information on the instances. Get power related information using the `-power` option.

Options and Arguments

file	Specifies the name of the file to which to write the report.
`-detail`	Reports information such as pin direction, propagated constants, propagated pins, and disabled timing arcs. Disabled timing arc info is only available for mapped designs.
instance	Specifies the leaf instance for which to generate the report.
`-power`	Reports instance leakage, internal power, net power and the computed probability, toggle rate, and net power on the nets of the instance, and the activity and power for each of the arcs. In case the switching activities are user-asserted, the values are appended with an asterisk (*). **Note:** The `lp_power_unit` attribute does not affect the units of the Arc Energy column.
`-timing`	Reports timing information, such as slew-in, load, slew-out, delay, and slack.

Example

- The following example reports timing information for the `n0000D3666` instance.

```
legacy_genus:/> report instance -timing n0000D3666

  ...
  Module:                dpldalgn
  ...
============================================================
         Instance Timing Info
         --------------------
 Instance n0000D3666 of libcell nor2

 Arc  Arc  Slew        Slew
From  To    In   Load  Out   Delay   Slack
--------------------------------------------
 A r  Y f  46.2  20.7  57.2  136.0  -1022.4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
     A f  Y r  46.2  20.7  57.2  136.0  -1022.4
     B r  Y f  16.5  20.7  57.2  134.0   -900.2
     B f  Y r  16.5  20.7  57.2  134.0   -900.2
```

- The following example shows the timing and power information for instance `o_m2_clk2_1_reg_0`.

```
============================================================
  ...
  Module:                 m1
  Technology library:     slow 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
============================================================
 Instance m2/o_m2_clk2_1_reg_0 of libcell EDFFX1

 Arc Arc Slew        Slew
From  To  In   Load   Out   Delay Slack
----------------------------------------
  CK   Q  0.0  10.0  173.9  387.0   inf
  CK  QN  0.0   0.0   64.3          inf


        Instance Power Info
        -------------------

 Instance m2/o_m2_clk2_1_reg_0 of Libcell EDFFX1

  Leakage    Internal      Net
 Power(nW)   Power(nW)  Power(nW)
----------------------------------
    32.822   20652.925    182.250

                          Computed          Computed           Net
Pin         Net         Probability    Toggle Rate(/ns)    Power(nW)
-------------------------------------------------------------------------
Q     o_m2_clk2_1[0]          0.6              0.0312        182.250
CK    n_0                     0.5              1.8724       4913.916
D     in_2[21]                0.5              0.0208        138.510
E     en_2[7]                 0.5              0.0208        473.850


 Arc  Arc                 Arc            Arc
From   To     When    Activity(/ns)   Energy(fJ)

    ------------------------------------------------
    CK     CK   !D & !E          0.4681        9.396
    CK     CK   D & !E           0.4681        9.563
    CK     CK   !D & E           0.4681       11.860
    CK     CK   D & E            0.4681       11.973
    D      D                     0.0208        7.726
    CK     CK   D & E            0.4681       11.973
    CK     Q                     0.0312        7.902
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows a detailed instance report.

```
legacy_genus:/>    report  instance  CG1  -detail
============================================================
    ...
    Module:                                    test
    Technology  library:          mylibraries
    Operating  conditions:      WCCOM  (balanced_tree)
    Wireload  mode:                    segmented
    Area  mode:                            timing  library
============================================================


                Instance  detail  information
                ----------------------------
Instance  CG1  of  libcell  /libraries/mylibraries/libcells/mycell

Pin  Direction  Constant    Clock
------------------------------
E      in
CP    in                                  clk(+)
TE    in                        0
Q      out                                clk(+)

Disabled  Arcs  by  constant  propagation
----------------------------------------
/libraries/mylibraries/libcells/mycell/inarcs/CP_TE_Ha0
/libraries/mylibraries/libcells/mycell/TE/inarcs/CP_TE_Sa0

Disabled  Arcs  by  instance  based  disable_timing
--------------------------------------------------


Disabled  Arcs  by  libcell  based  disable_timing
--------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report memory

`report memory [> file]`

Reports the memory resource used by the compiler in the computing platform.

Options and Arguments

file	Specifies the name of the file to which to write the report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report memory_cells

`report memory_cells [library] [> file]`

Reports the memory cells in the library.

Options and Arguments

file	Specifies the name of the file to which to write the report.
library	Specifies the name of the library for which to report the memory cells. If no library is specified, the report applies to all libraries that are loaded.

Example

The following command lists the memory cells in the RF32X32_lib library.

```
============================================================================
============================================================================

         Library : RF32X32_lib
         Memory Cell: RF32X32
         Memory Type: ram
         Memory address width: 5
         Memory word width: 32
-------------------------------------------------------------------------------------------------------------
   PIN/BUS       DIRECTION IS_BUS             BUS_TYPE CAPACITANCE RELATED_PIN     TIMING_TYPE    TIMING_SENSE
-------------------------------------------------------------------------------------------------------------

         Q  output 31 to 0    YES         RF32X32_DATA                     CLK     rising_edge       non_unate

         D   input 31 to 0    YES         RF32X32_DATA       0.002         CLK     hold_rising
                                                                           CLK    setup_rising


       CEN    input 0 to 0     NO                            0.002         CLK     hold_rising
                                                                           CLK    setup_rising

       WEN    input 0 to 0     NO                            0.010         CLK     hold_rising
                                                                           CLK    setup_rising

         A    input 4 to 0    YES      RF32X32_ADDRESS       0.008         CLK     hold_rising
                                                                           CLK    setup_rising
-----------------------------------------------------------------------------------------------------------
     CLOCK       DIRECTION CAPACITANCE MAX_TRANSITION MIN_PULSE_WIDTH_HIGH  MIN_PULSE_WIDTH_LOW  MIN_PERIOD
-----------------------------------------------------------------------------------------------------------

       CLK    input 0 to 0       0.038          1.000                0.055                0.130      0.867
-----------------------------------------------------------------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report messages

```
report messages [-all] [-include_suppressed] [-error]
     [-warning] [-info] [-message_list object...] [> file]
```

Summarizes the information, warning and error messages that have been issued by Genus in the current run since the last report. The report contains the number of times the message has been issued, the severity of the message, the identification number, and the message text.

The `-all` option does not report those messages that were suppressed through the `suppress_messages` command. Specify the `-include_suppressed` option to report such messages.

By adding report messages to your Tcl prompt, Genus can summarize all messages issued during the last command right before prompting you for more input. This is useful after long commands (such as `elaborate`) that can generate many messages.

Info type messages print is driven by an attribute: information_level. The higher number (between 1 and 11, default 1) the more information messages are printed out. A high number will make the log file bigger but it will contain more information useful to analyze and debug.

Like many objects in the database, messages have associated attributes. Some attributes can be used to drive the way the messages are printed out, save in the logs etc.

Options and Arguments

`-all`	Returns a list of all issued message IDs as a Tcl result. The result does not include suppressed message IDs.
`-error`	Returns a list of issued error message IDs as a Tcl result.
`file`	Specifies the name of the file to which to write the report.
`-include_suppressed`	Returns a list of suppressed message IDs as a Tcl result. These messages were suppressed through the `suppress_messages` command.
`-info`	Returns a list of issued information message IDs as a Tcl result.
`-message_list` *`object`*	Returns a list of issued message IDs as specified in the list.
`-warning`	Returns a list of issued warning message IDs as a Tcl result.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

Note: The following examples all apply to the same Genus session.

- The following example is the first request to report messages in a session.

```
legacy_genus:/> report messages

  =================
   Message Summary
  =================

Num  Sev      Id                        Message Text
---------------------------------------------------------------------------
  1 Info ELAB-VLOG-9 Variable has no fanout.
                     This variable is not driving anything and will be
                     simplified
  3 Info LBR-30      Promoting a setup arc to recovery.
                     Setup arcs to asynchronous input pins are not
                     supported
  3 Info LBR-31      Promoting a hold arc to removal.
                     Hold arcs to asynchronous input pins are not
                     supported
  1 Info LBR-54      Library has missing unit.
                     Current library has missing unit.
```

- The following example executes the `report messages` command immediately after the previous `report messages` command and consequently does not return any new messages.

```
legacy_genus:/> report messages
```

- The following example is the third `report messages` command in a row, but because the `-all` option is specified, the same output as with the first command is given.

```
legacy_genus:/> report messages -all

  =================
   Message Summary
  =================

Num  Sev      Id                        Message Text
---------------------------------------------------------------------------
  1 Info ELAB-VLOG-9 Variable has no fanout.
                     This variable is not driving anything and will be
                     simplified
  3 Info LBR-30      Promoting a setup arc to recovery.
                     Setup arcs to asynchronous input pins are not
                     supported
  3 Info LBR-31      Promoting a hold arc to removal.
                     Hold arcs to asynchronous input pins are not
                     supported
  1 Info LBR-54      Library has missing unit.
                     Current library has missing unit.
```

- The following example requests to print all error messages since this run was started, but no error messages were found:

```
legacy_genus:/> report messages -all -error
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related attributes	count
	max_print
	print_count
	screen_max_print
	screen_print_count

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report min_pulse_width

```
report min_pulse_width
     {pin|port}...
```

Prints the list of pins and ports in the design where the minimum clock pulse width is violated.

Options and Arguments

`{pin|port}`	Pin or port where the minimum clock pulse width is violated.

Related Information

Related commands:	create_clock on page 1410
	set_min_pulse_width on page 1616

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report mode

```
report mode
     instance [ > file]
```

Reports the active and inactive modes of the specified instance.

Note: You can activate or deactivate the library modes using the `set_mode` SDC command.

Options and Arguments

file	Specifies the name of the file to which to write the report.
instance	Specifies the instance for which to report the mode information,

Example

The following command shows the mode information for instance `i1`.

```
legacy_genus:/> report mode i1
 ------------------------------
  Mode Group setup of i1
 ------------------------------
   Mode Name     Status
-------------------------
 slavePushout     ACTIVE
 masterCapture  Inactive
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report module

```
report module
     [-depth integer] [-min_count integer]
     [-summary] [module]...
```

Prints a report for the specified modules.

Options and Arguments

`-depth` *`integer`*	Specifies the number of levels of recursion.
`-min_count` *`integer`*	Specifies the minimum instance count per module.
`module`	Specifies the module to report on.
`-summary`	Specifies to report the area summary.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report multibit_inferencing

```
report multibit_inferencing [-comb] [-seq] [-power]
     [[-instance_hier instance] [design] | -lib]
     [-not_merged_summary] [-not_merged_instance_list]
     [-reason_not_merged string] [-tcl]
     [-mapping] [-no_header] [-iso_ls] > file
```

Reports detailed information on the combinational and sequential multibit cells that are used in the design or available in the libraries. Use the options to filter the information. When no options are specified, the report returns the information for all multibit cells used in the design.

Options and Arguments

`-comb`
: Limits the report to combinational multibit cells.

  For each multibit libcell the following is reported: bitwidth, the number of instances in the design mapped to this libcell, total area of these instances, the multibit conversion ratio (number of single bit instances merged to this multibit cell to the total number of single bit instances), and the library.

design
: Specifies the design for which to create the report.

  If no design is specified, the report is given for the current design.

  This option is required when multiple designs are loaded.

  Note: You cannot specify this option together with the `-lib` option.

file
: Redirects the report to the specified file.

`-instance_hier` *instance*
: Prints the report for the specified hierarchical instance.

`-iso_ls`
: Limits the report to iso/ls multibit cells.

`-lib`
: Reports the multibit cells available in the libraries.

  For each libcell, the following is reported: value of the avoid attribute, usability of the multibit cell, bitwidth, cell area, cell leakage power, library, library domain.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`Multibit_Usable` shows whether a libcell is valid to use during multibit merging or not.

Note: This option can be used after the libraries are loaded and before the design is loaded.

Note: You cannot specify this option together with the `-instance_hier` option or when you specify the design.

`-mapping` Reports the mapping between any multibit instance and the corresponding original single-bit instances.

`-no_header` Prints the report without a header.

`-not_merged_instance_list`

Prints all single bit sequential instances that could not be merged into multibit instances.

`-not_merged_summary` Prints for each of the different reasons the number of instances that were not merged into a multibit instance.

`-power` Reports leakage power information for the multibit cells in the design.

`-reason_not_merged` *`string`*

Reports the instances that could not be merged for the specified reason. Specify any of the reasons listed in Reasons for Not Merging Instances, or specify "all" to report all instances with the reason why they could not be merged.

`-seq` Limits the report to sequential multibit cells.

For each multibit libcell the following is reported: bitwidth, the number of instances in the design mapped to this libcell, total area of these instances, the multibit conversion ratio (number of single bit instances merged to this multibit cell to the total number of single bit instances), and the library.

`-tcl` Generates a `create_multibit.tcl` file which contains a list of commands to merge single bit flops individually.

Reasons for Not Merging Instances

INST_DFT_PRESERVE

These instances are part of a scan chain or scan segments.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

By default, instances that are part of a scan chain are merged (scan chain merging). In some cases, certain limitations still prevent merging within a scan chain.

INST_DONT_INFER

The `dont_infer_multibit` attribute is set to `true` on this instance. This setting causes this single bit instance to be skipped during multibit merging.

INST_MISMATCH_EXCEPTION

Only instances with the same timing exceptions can be merged into multibit instances. These instances have a mismatch in one or more of the exceptions, which excludes them from multibit merging.

INST_MISMATCH_SPRG_CONSTRAINTS

Only instances with the same SRPG constraints can be merged into multibit instances. These instances have a mismatch in one or more of the constraints which excludes them from multibit merging.

INST_MULTIPLE_CONNECTED_OUTS

These instances have an output pin that is neither a q, qb, or a scan-data output pin and the pin is driving some load.

The tool only recognizes q, qb, or scan-data output pins for multibit cell inferencing (MBCI). These instances have output pins that are not recognizable which excludes them from multibit merging.

INST_NET_PRESERVE

A net connected to the instances is preserved. By default, multibit merging is allowed on instances with preserved nets. In this case, the `multibit_preserved_net_check` was set to `true` to prevent multibit merging when any of the nets connected to the instances is marked preserved.

INST_ NO_CLUSTER

During physical multibit merging, instances that are closer are merged into multibit instance clusters. These instances do not belong to any cluster and were rejected for multibit merging.

INST_NO_VALID_REPLACEMENT

No multibit libcell was found in the library that is compatible with the single bit instances.

For example, the single bit instances are scan instances where as all multibit instances in

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

the library are non-scan.

INST_ONLY_ONE_BIT

The module has either only one single bit instance which cannot be converted into a multibit instance since two instances are needed for creating multibit, or this is the only instance that remains after creating multibit instances from the single bit instances in this module.

INST_OUTPUT_PIN_BUNDLED / INST_OUTPUT_PIN_BUSSED

Single bit instances with bundled or bussed pins are not considered for multibit merging.

INST_PIN_CASE_CONSTANT

If any instance pin is forced to assume a constant value for timing analysis purposes, that instance is not considered for multibit merging.

INST_PRESERVE

The instance or any of its pins is preserved. Preserved instances are not merged into multibit instances.

INST_SCAN_OUT_NON_LOOP

If a scan flip-flop is not part of a scan chain, the scan-data output of the instance should be connected in loopback mode (scan-data output pin of the scan flop is connected to its own scan-data input pin). In this case, the scan-data output of the instance is not connected in loopback mode, therefore the instance is not considered for multibit merging.

INST_SPLIT

This single bit instance was created by splitting a multibit instance for timing purposes. Merging it again into a multibit instance would not benefit the qor.

INST_TIMING_EXCEPTION

The instances have a timing exception. By default, these instances are considered for multibit merging. In this case, the `multibit_auto_exclude_registers_with_exceptions` attribute was set to `true` to prevent multibit merging of sequential instances with exceptions.

INST_UNRECOGNISED_CONTROL_INPUT

The instances have a pin that is not recognized by the synthesis tool. Instances with pins that are not recognizable are not considered for multibit merging.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

INST_WORSE_QOR

Adding this single bit instance to a multibit instance would worsen the qor.

Examples

- The following command reports the multibit cells in the libraries.

```
legacy_genus:/> report multibit_inferencing -lib
=========================================================
  Generated by:           Genus Synthesis Solution version
  Generated on:           date
  Technology libraries:   XYZ_primitive 1.005489
                          XYZ_multibit 0.000001
  Operating conditions:   WORST_TREE (worst_case_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=========================================================

 Library : Combinational Multibit Libcells info
==========================================================================================

Comb_Mbit libcell  Avoid  Bitwidth   Leakage Power    Area                Library                Library Domain
------------------------------------------------------------------------------------------------------------------
AND2_2B            false         2            1.23    1.10                XYZ_multibit           NA


Seq_Mbit libcell  Avoid    Multibit_Usable  Bitwidth   Leakage Power    Area                Library           Library Domain
------------------------------------------------------------------------------------------------------------------------------
DFF_1_2B          false              true         2           14.32    7.53             XYZ_multibit         lp_domain
DFF_1_4B          false              true         4           14.32   15.05             XYZ_multibit         lp_domain
DFF_1_8B          false             false         8           14.32   30.11             XYZ_multibit         lp_domain

. . .
DFF_1_2B          false              true         2           14.32    7.53             XYZ_multibit         gp_domain
DFF_1_4B          false              true         4           14.32   15.05             XYZ_multibit         gp_domain
DFF_1_8B          false             false         8           14.32   30.11             XYZ_multibit         gp_domain
```

- The following command reports the combinational multibit cells used in the design.

```
legacy_genus:/> report multibit_inferencing -comb
=============================================================
...
=============================================================

Combinational Multibit cells usage statistics
==================================================================================================================
Total Combinational instances merged: 2
Total Combinational instances not merged: 0

Comb_Mbit libcell  Bitwidth  Count   Total Area Multibit Conversion %              Library
------------------------------------------------------------------------------------------------------------------
AND2_2B                    2      1         1.10         100.00                     XYZ_multibit
------------------------------------------------------------------------------------------------------------------
Total                             1         1.10         100.00
```

- The following command reports the combinational multibit cells used in the design with their leakage power information.

```
legacy_genus:/> report multibit_inferencing -comb -power
=========================================================
...
=========================================================

Combinational Multibit cells usage statistics
==================================================================================================================
Total Combinational instances merged: 2
Total Combinational instances not merged: 0

Comb_Mbit libcell  Bitwidth  Count   Total Area    Total Power(nW)    Multibit Conversion %           Library
------------------------------------------------------------------------------------------------------------------
AND2_2B                    2      1         1.10          1.23                 100.00                  XYZ_multibit
------------------------------------------------------------------------------------------------------------------
Total                             1         1.10          1.23                 100.00
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports the sequential multibit cells used in the design, along with individual multibit conversion percentage for latches as well as for flip-flops.

```
legacy_genus:/> report multibit_inferencing -seq
=========================================================
  ...
=========================================================

Sequential Multibit cells usage statistics
==========================================================================================

                          Merged        Not Merged     Multibit Conversion %
-------------------------------------------------------------------------
All Sequentials               14                0           100.00
-FlipFlops                    14                0           100.00
-Latches                       0                0             0.00
-------------------------------------------------------------------------


Seq_Mbit libcell      Bitwidth  Count   Total Area Multibit Conversion %              Library
-----------------------------------------------------------------------------------------------
DFF_1_2B    (F)              2      1         7.53         14.29                      XYZ_multibit
DFF_1_4B    (F)              4      1        15.05         28.57                      XYZ_multibit
DFF_1_8B    (F)              8      1        30.11         57.14                      XYX_multibit
-----------------------------------------------------------------------------------------------
Total                                3        52.69        100.00
```

- The following command lists the number of instances that did not get merged per reason.

```
legacy_genus:/> report multibit_inferencing -not_merged_summary -no_header

Info  : Reasons for not merging to multibit could get overwritten in multiple iterations of incremental optimization
      : and the same captured below are specific to the last iteration.
      : Recommendation is to run this command after each call of incrementaloptimization.


Sequential Multibit not merged statistics
==========================================================================================
Total Sequential instances not merged: 1318

  Reason of not merging         Sequential Instances                        Description
-----------------------------------------------------------------------------------------
INST_NO_VALID_REPLACEMENT                      1312     No valid replacement /No valid multibit libcell present
INST_SPLIT                                        5     Single bit created after splitting the multibit
INST_WORSE_QOR                                    1     Move not accepted as it is giving worse QoR
-----------------------------------------------------------------------------------------
```

- The following command lists the instances that did not get merged with the reason.

```
legacy_genus:/> report multibit_inferencing -not_merged_instance_list -no_header
Info  : Reasons for not merging to multibit could get overwritten in multiple
iterations of incremental optimization
      : and the same captured below are specific to the last iteration.
      : Recommendation is to run this command after each call of incremental
optimization.

Sequential Multibit not merged statistics
=================================================================
Total Sequential instances not merged: 1318

Total Unmerged Sequential Instances
----------------------------------
a_reg[0]
a_reg[1]
a_reg[0]
a_reg[1]
b_reg[0]
...
...
f_reg[3][7]
f_reg[3][8]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
f_reg[3][9]
---------------------------------
...
```

- The following command shows the mapping between any multibit instance to the corresponding original single bit instances.

```
legacy_genus:/> report multibit_inferencing -mapping -no_header

Sequential Multibit Mapping Report
==========================================================================
Total Multibit Sequential instances : 3

     Multibit instances     Single bit Instance Mapping
--------------------------------------------------------------------------

s/d_hier/CDN_MBIT_d_reg[0]_MB_d_reg[1]
  CDN_MBIT_d_reg[0]_MB_d_reg[1]/Q0     d_reg[0]/Q
  CDN_MBIT_d_reg[0]_MB_d_reg[1]/Q1     d_reg[1]/Q

s/d_hier/CDN_MBIT_d_reg[2]_MB_d_reg[3]
  CDN_MBIT_d_reg[2]_MB_d_reg[3]/Q0     d_reg[2]/Q
  CDN_MBIT_d_reg[2]_MB_d_reg[3]/Q1     d_reg[3]/Q

s/CDN_MBIT_b_reg[2]_MB_b_reg[3]
  CDN_MBIT_b_reg[2]_MB_b_reg[3]/Q0     b_reg[2]/Q
  CDN_MBIT_b_reg[2]_MB_b_reg[3]/Q1     b_reg[3]/Q
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report net_cap_calculation

```
report net_cap_calculation net
     [-fall] [-rise] [> file]
```

Reports the capacitance values for the different pins or ports that are connected to the specified net.

- For a pin, the capacitance value is the capacitance of the libcell pin (obtained from the `.lib`).
- For a port, the capacitance value is any capacitance annotated on the port (sum of the `external_pin_cap` and `external_wire_cap` attributes).
- The net capacitance is computed from the wire-load model available in the library. This is based on the `wireload_mode` attribute (`top`, `segmented`, or `enclosed`).
- The final capacitance is the sum of the net capacitance and the pin and port capacitance values to which the net is connected.

The columns "Terms from .lib" and "Cap from .lib" in the report will be populated if the wireload model in the `.lib` appears as a table. Otherwise, it will be empty.

This command is not supported on those nets that have the `physical_cap` attribute set on them. Also, when you are in PLE mode, only the final capacitance is shown (no computation).

Options and Arguments

file	Redirects the report to the specified file.
net	Specifies the net name for which the report should be generated.

Related Information

Related command: report net_res_calculation on page 660

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report net_delay_calculation

```
report net_delay_calculation
     [-driver_pin {port|pin}...]
     [-load_pin {port|pin}...]
     [-rise_rise] [-fall_fall] [> file]
```

Reports the net delay, in picoseconds, between the specified driver and load pins. Both the driver and load pins should be on the same net. The delay computed would depend upon the tree type used (`tree_type` attribute): best case, worst case, or balanced tree.

Note: This command assumes that the `interconnect_mode` attribute is set to `wireload`.

Options and Arguments

`-driver_pin {port|pin}`
: Specifies the starting port or pin on which to obtain the net delay.

`-fall_fall`
: Restricts report of net delay to fall to fall arcs

file
: Redirects the report to the specified file.

`-load_pin {port|pin}`
: Specifies the ending port or pin on which to obtain the net delay.

`-rise_rise`
: Restricts report of net delay to rise to rise arcs.

Example

- The following command provides a report based on the in1[0] driver pin.

```
legacy_genus:/designs/areid/ports_in> report_net_delay_calculation -driver_pin
in1[0]
============================================================
...
Operating conditions:   slow (balanced_tree)
Wireload mode:          segmented
============================================================

Formula: (Wres/f) * (Pcap + Wcap/f)

From      To        Wire res  Wire cap Fanout Pin cap of  Total pin cap of   Net
pin       pin        of net    of net of net   to pin       all to pins    delay

-------------------------------------------------------------------------------
in1[0]    inst1/g43/A  0.000      7.2    1      5.3             5.3         0.0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by this attribute: tree_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report net_res_calculation

`report net_res_calculation net [> file]`

Reports the total wire resistance of the specified net. The total wire resistance is the sum of the individual net segment resistance (obtained from the wire-load model) and the external wire resistance (annotated on any port and obtained from the `external_wire_res` attribute) that is connected to the net.

- For a port, the resistance value is any resistance annotated on the port (the `external_wire_cap` attributes).
- The net resistance is computed from the wire-load model available in the library. This is based on the `wireload_mode` attribute (`top`, `segmented`, or `enclosed`).
- The final resistance is the sum of the net resistance and the pin and port resistance values to which the net is connected.

The columns "Terms from .lib" and "Cap from .lib" in the report will be populated if the wire-load model in the `.lib` appears as a table. Otherwise, it will be empty.

This command is not supported on those nets that have the `physical_res` attribute set on them. Also, when you are in PLE mode, only the final resistance is shown (no computation).

Options and Arguments

file	Redirects the report to the specified file.
net	Specifies the net name for which the report should be generated.

Related Information

Related command: report net_cap_calculation on page 657

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report nets

```
report nets [-hierarchical] [-pin pin...]
     [-minfanout integer] [-maxfanout integer]
     [net | instance]... [-sort string]
     [-cap_worst integer] [> file]
```

Generates a report on the nets of the current design. The report gives information for the top-level nets in the design. You can specify pin names, nets, instances, maximum and minimum fanout threshold values, nets, and instances. Control the data printed out using the `-minfanout` and `-maxfanout` options for nets that have fanout between these values.Nets that are followed by the "`@`" symbol in parenthesis indicate SPEF annotation.

Options and Arguments

`-cap_worst` *`integer`*	Specifies the number of worst capacitance nets that are to be reported.
`file`	Specifies the name of the file to which to write the report.
`-hierarchical`	Reports all the nets in the design hierarchy.
`-maxfanout` *`integer`*	Specifies an *`integer`* value and reports nets whose fanouts are below the given threshold value.
`-minfanout` *`integer`*	Specifies an *`integer`* value and reports nets whose fanouts are above the given threshold value.
`net``|`*`instance`*	Reports information on the specified nets or nets belonging to the instance.
`-pin` *`pin`*	Specifies a list of pin names and reports the nets connected to the pins.
`-sort`	Specifies the field name on which to sort. Valid field names are `load`, `resistance`, or `capacitance`.

Examples

- The following example shows that the `address` nets are SPEF annotated while the `accum` nets are not.

```
============================================================
...
============================================================
address[0] (@)       1         1     1.0      0.000
address[1] (@)       1         1     0.8      0.000
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
accum[0]             1         1     2.1      0.000
accum[1]             1         1     8.5      0.000
```

- The following example reports the five worst capacitance nets in the top level.

```
legacy_genus:/> report nets -cap_worst 5
============================================================
  Generated by:           Genus Synthesis Solution Version
  Generated on:           Date
  Module:                 m1
  Technology library:     slow 1.5
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          segmented
============================================================
                          Wire      Wire   Wireload
  Net   Loads   Drivers Cap(fF) Res(k-ohm)    Model
----------------------------------------------------
ai          2         3    14.5      0.000
n_135       2         2    10.4      0.000
n_0         2         1     6.7      0.000
ao[5]       1         2     6.7      0.000
bi          1         1     3.6      0.000
============================================================
```

- The following example sorts the nets in the top level by the number of loads.

```
legacy_genus:/> report nets -sort load
============================================================
...
============================================================
                          Wire      Wire   Wireload
  Net   Loads   Drivers Cap(fF) Res(k-ohm)    Model
----------------------------------------------------
n_1         8         1
clk         3         1     0.0      0.000
n_135       2         2    10.4      0.000
n_0         2         1     6.7      0.000
ai          2         3    14.5      0.000
......
============================================================
```

- The following example provides specific information on the `n_0 ai` net.

```
legacy_genus:/> report net n_0 ai
============================================================
...
============================================================
     Total   Slew Slew
Net Cap(fF)  Rise Fall Driver(s)   Load(s)
--------------------------------------------
n_0    20.3   0.0  0.0  g24/Y    g23/A
                                 g22/A
                                 bx_reg3/CK
ai     12.0   0.0  0.0  ai       g25/A
                                 bx_reg2/D
============================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example reports all the nets in the top level of the current design whose fanout is less than 2.

```
legacy_genus:/> report net -maxfanout 2
================================================================
...
================================================================
                                  Wire      Wire     Wireload
      Net       Loads   Drivers Cap(fF) Res(k-ohm)     Model
-----------------------------------------------------------------
in1a[0]             1         1     0.4      0.000    AL_SMALL
in1a[1]             1         1     0.4      0.000    AL_SMALL
in1b[0]             1         1     0.4      0.000    AL_SMALL
in1b[1]             1         1     0.4      0.000    AL_SMALL
out2a[0]            1         1     0.4      0.000    AL_SMALL
out2a[1]            1         1     0.4      0.000    AL_SMALL
out2b[0]            1         1     0.4      0.000    AL_SMALL
out2b[1]            1         1     0.4      0.000    AL_SMALL
out3a[0]            1         1     0.0      0.000    AL_SMALL
out3a[1]            1         1     0.0      0.000    AL_SMALL
out3b[0]            1         1     0.0      0.000    AL_SMALL
out3b[1]            1         1     0.0      0.000    AL_SMALL
out4a[0]            1         1     0.0      0.000    AL_SMALL
out4a[1]            1         1     0.0      0.000    AL_SMALL
out4b[0]            1         1     0.0      0.000    AL_SMALL
out4b[1]            1         1     0.0      0.000    AL_SMALL
=================================================================
```

You can specify an instance name to the above example and get information on the nets associated with the instance(s) whose fanout is less than 2.

- The following example reports the nets associated with the `g22/A bx_reg2/D` pin.

```
legacy_genus:/> report net -pin "g22/A bx_reg2/D"
==============================================================
...
==============================================================
     Total   Slew Slew
Net Cap(fF)  Rise Fall Driver(s)   Load(s)
--------------------------------------------------------------
n_0    20.3   0.0  0.0  g24/Y    g23/A
                                 g22/A
                                 bx_reg3/CK
ai     12.0   0.0  0.0  ai       g25/A
                                 bx_reg2/D
==============================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report opcg_equivalents

```
report opcg_equivalents
     [-pinVal] [design] [> file]
```

Reports the OPCG-equivalence mappings specified using the `set_opcg_equivalent` command(s).

Options and Arguments

design	Specifies the design for which you want to report the OPCG-equivalence mappings.
file	Specifies the name of the file to which to write the report.
-pinVal	Prints the pin constant values.

Example

The following report shows two OPCG-equivalency mappings. For each mapping, the report lists the scan cell, the equivalent OPCG cell, the edge-mode pin, the loopback pin, and the pin mappings between the two cells.

```
legacy_genus:/> report opcg_equivalents
OPCG equivalent table contains 2 entries {
{
    Scan cell: /libraries/myspecial/libcells/SDFFQN_X1M
    Opcg cell: /libraries/myspecial/libcells/S2DFFQQN_X1M
    Edge_mode: TEL
    Loop_back: TI
    Pin map:  {SE SE} {SI SI} {D D} {CK CK} {QN QN}
  }

  {
    Scan cell: /libraries/myspecial/libcells/SDFFQ_X1M
    Opcg cell: /libraries/myspecial/libcells/S2DFFQQN_X1M
    Edge_mode: TEL
    Loop_back: TI
    Pin map: {D D} {CK CK} {Q Q} {SE SE} {SI SI}
  }
}
```

Related Information

Affected by these commands: reset_opcg_equivalent on page 1138

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_opcg_equivalent on page 1142

Related commands: replace_opcg_scan on page 1121

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report ple

`report ple [design]... [> file]`

Returns the physical layout estimation information for the specified design. The command reports information like aspect ratio, shrink factor, site size, layer names, direction of layers (**H**orizontal, **V**ertical or **U**ndefined), layer utilization, capacitance, resistance, area, and the source used to extract the physical information.

Options and Arguments

design	Specifies the design name on which to report. If no design is specified, the current design is loaded.
file	Specifies the name of the file to which to write the report.

Example

- The following example reports the ple information for the current design.

```
    legacy_genus:/> report ple
============================================================
  Generated by:           Genus Synthesis Solution version
  Generated on:           date
  Module:                 DTMF_CHIP
  Technology libraries:   tsmc18 1.0
                          tpz973g 230
                          pllclk 4.3
                          ram_128x16A 1.1
                          ram_256x16A 1.1
                          rom_512x16A 1.1
                          physical_cells
  Operating conditions:   slow
  Interconnect mode:      global
  Area mode:              physical library
============================================================

Aspect ratio        : 1.00
Shrink factor       : 1.00
Scale of res/length : 1.00
Scale of cap/length : 1.00
Net derating factor : 1.00
Site size           : 5.70 um (from lef [tech+cell])

                                  Capacitance
  Layer                            / Length         Data source:
Name        Direction Utilization  (pF/micron)       cap_table_file
-----------------------------------------------
M1              H         1.00        0.000274
M2              V         1.00        0.000242
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
M3              H         1.00        0.000242
M4              V         1.00        0.000242
M5              H         1.00        0.000242
M6              V         1.00        0.000304

                                     Resistance
  Layer                               / Length         Data source:
Name        Direction Utilization  (ohm/micron)       lef_library
-------------------------------------------------
Metal1          H         1.00         0.439130
Metal2          V         1.00         0.360714
Metal3          H         1.00         0.360714
Metal4          V         1.00         0.360714
Metal5          H         1.00         0.360714
Metal6          V         1.00         0.102273

                                       Area
  Layer                               / Length        Data source:
Name        Direction Utilization     (micron)        lef_library
-------------------------------------------------
Metal1          H         1.00         0.230000
Metal2          V         1.00         0.280000
Metal3          H         1.00         0.280000
Metal4          V         1.00         0.280000
Metal5          H         1.00         0.280000
Metal6          V         1.00         0.440000
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report port

`report port [-delay] [-driver] [-load] port... [> file]`

Generates reports on the ports of the current design. By default, the report gives information on port direction, external delays, exception objects and their types, driver, slew, fanout load, pin capacitance and wire capacitance for the ports. You can also specify the port names on which the report is to be generated and control the data printed using the `-delay`, `-driver` and `-load` options.

Options and Arguments

`-delay`	Reports external delay information (rise and fall delay and the external delay object)
`-driver`	Reports the external driver name and the slew (rise and fall) values of the ports.
`file`	Specifies the name of the file to which to write the report.
`-load`	Reports the external fanout load and the pin and wire capacitance values of the ports.
`port`	Specifies the port for which to generate the report.

Example

The following example reports the external delay information on the `ck1`, `e_out[6]`, and `ena` ports.

```
legacy_genus:/> report port -delay -driver -load ck1 e_out[6] ena
External Delays & Exceptions
----------------------------
                          Rise   Fall  Ext Delay      Exception
   Port     Dir  Clock    Delay  Delay   Object      Object/Type
-------------------------------------------------------------------

 ck1       in     CLK1   700.0  700.0  in_del_1                 N/A
                  CLK2   500.0  500.0  in_del_2

 e_out[6]  out    CLK1   200.0  200.0  ou_del_1  del_1 (path_delay)
                  CLK2   300.0  300.0  ou_del_2
                         300.0  300.0  outrxtl

 ena       inout  CLK1   700.0  700.0  in_del_1                 N/A
                  CLK2   500.0  500.0  in_del_2

 ena       inout  CLK1   200.0  200.0  ou_del_1  del_1 (path_delay)
                  CLK2   300.0  300.0  ou_del_2
                         300.0  300.0  outrxtl
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power

```
report power
     { -rtl_cross_reference [-detail]
       [-flat [-nworst number]] [-sort mode]
       [design | instance]...
       [-mode mode | -power_mode power_mode] [-tcf_summary]
     | [-hier | -flat [-nworst number]] [-depth number]
       [-sort mode] [design | instance | net | pin]...
       [-mode mode | -power_mode power_mode] [-tcf_summary]
     | -clock_tree [clock]...
       -width float -height float }
     [-verbose] [> file]
```

Reports the power consumed. The information returned depends on your current *position* in the design hierarchy and on the specified objects. If no objects are specified, the report is given for the design or instance at the current *position* in the design hierarchy.

With the `-rtl_cross_reference` option specified, the power consumed by the instances is cross-referenced to the corresponding line in the RTL files. Set the `hdl_track_filename_row_col` attribute to `true` before elaboration to enable filename, column, and line number tracking.

Note: Nets connected to primary inputs and outputs are only reported at the top-level of an instance-based power report.

Options and Arguments

clock	Specifies the name of a clock for which you want to estimate the clock tree power. If no clock is specified, the power is estimated for all clocks in the design.
`-clock_tree`	Estimates the power of the clock tree. You can use this option with generic and mapped netlists. **Note:** This option is only available with the `Genus_Low_Power_Opt` License Key.
`-depth` *number*	Specifies the number of hierarchy levels to descend in the report. If an instance is specified, the depth starts from the position of that instance in the design hierarchy. Use a non-negative integer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This option applies to instance-based power reports, but is not supported with the `-rtl_cross_reference` option.

Default: `infinite` (all levels of the hierarchy)

`design`

Specifies the design for which you want the power to be reported. Specify the path name to the design.

If your current *position* in the design hierarchy is at the design level and you have only one design loaded, the report is by default given for the design.

`-detail`

Adds an abbreviated version of the RTL line and a list of the instances that correspond to that RTL line. In this report, the dynamic power is replaced with the internal power and net power (the two components of dynamic power). The detailed report also returns the power information for the primary inputs.

Note: This option only applies if you specified the `-rtl_cross_reference` option.

`file`

Specifies the name of the file to which to write the report.

`-flat`

Reports the power of leaf instances (combinational and sequential instances) starting from the *current* position in the hierarchy.

- If you perform a gate-level power analysis, the number of hierarchical levels that are expanded depends on the setting of the `-depth` option.
- If you perform RTL power analysis using the `-rtl_cross_reference` option, power information for all modules in the current hierarchy is shown.

Note: This option only applies to instance-based power reports.

`-full_instance_names`

Reports the full path names of the instances.

`-height` *`float`*

Specifies the estimated chip height (in microns).

Note: This option can only be specified with the `-clock_tree` option and is optional if a DEF file was read in.

`-hier`

Reports the power of hierarchical instances. The number of hierarchical levels shown depends on the setting of the `-depth` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you specify neither the `-flat` or `-hier` option, the Genus-LP engine uses the `-hier` option by default.

Note: This option applies to instance-based power reports, but is not supported with the `-rtl_cross_reference` option.

instance

Specifies the instance for which you want the power to be reported. Specify the path name to the instance.

`-mode` *mode*

Only prints the power report for the specified timing mode.

In this case, only the clocks in the specified timing mode are considered in the power calculations. If no mode is specified, all clocks are considered.

net

Specifies the net for which you want the power to be reported. Specify the path name to the net.

Note: Net-based power reports are not supported with the `-rtl_cross_reference` option.

`-nworst` *number*

Prints only the top worst entries of a sorted report.

This option applies only to instance-based reports, and can only be used with the `-flat` option.

pin

Specifies the pin for which you want the power to be reported. Specify the path name to the pin.

`-power_mode` *power_mode*

Only prints the power report for the specified power mode.

The specified power mode must correspond to one of the power mode names defined with a `create_power_mode` command in the CPF file that was read in.

If you have a multi-mode design and you omit this option, the report will apply to the current state of the design.

Note: This option cannot be used with the `-clock_tree` option.

`-rtl_cross_reference`

Cross-references the power consumed to the corresponding line in the RTL files. The report also returns the leakage power, dynamic power, and total power for the top-level design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-sort` *`mode`*

Indicates how to sort the report.

The following modes are available for *instance*-based reports:

`dynamic`
: Sorts by descending total dynamic power, which is the sum of the internal and net power.

  Note: If you specified the `-rtl_cross_reference` option, this mode is available without any other options or with the -`verbose` option.

`file`
: Sorts by RTL file and line number.

  Note: This option applies only to RTL power analysis and is the default for RTL power analysis.

`internal`
: Sorts by descending internal power.

  Note: If you specified the `-rtl_cross_reference` option, this mode is only available if you also specified the `-detail` or `-verbose` option.

`leakage`
: Sorts by descending leakage power.

`net`
: Sorts by descending net power.

  Note: If you specified the `-rtl_cross_reference` option, this mode is only available if you also specified the `-detail` or `-verbose` option.

`total`
: Sorts by descending total power.

  Note: If you specified the `-rtl_cross_reference` option, this mode is available without any other options or with the -`verbose` option.

The following modes are available for *net*-based reports:

`dynamic` (default)
: Sorts by descending total dynamic power.

`load`
: Sorts by descending capacitive load on the net.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`net`	Sorts by descending total switching power.
`prob`	Sorts by descending probability of the nets being high.
`rate`	Sorts by descending toggle rates of the nets.

Note: If a report is requested for nets and instances, but the specified sort mode applies only to one category, the other category will be sorted according to its default.

`-tcf_summary`

Adds a summary to the power report listing the following:

- The number of primary inputs asserted in the design.
- The total number of primary inputs connected in the design.
- The number of sequential outputs asserted in the design.
- The total number of sequential outputs connected in the design.
- The total number of nets in the design.
- *Nets asserted* refer to nets with asserted switching activities (probability and toggle rate).
- *Asserted clock nets* refer to nets whose `lp_probability_type` or `lp_toggle_rate_type` attribute value is set to `clock`.
- *Constant nets* refer to nets whose driver is either a constant object `0` or `1`.

For net-based reports, an asterisk (*) is appended to each net that has user-asserted switching activities.

`-verbose`

Replaces the dynamic power column with the components of the dynamic power—the internal power and net power.

`-width` *`float`*

Specifies the estimated chip width (in microns).

Note: This option can only be specified with the `-clock_tree` option and is optional if a DEF file was read in.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command requests a basic RTL power analysis of design `mult_bit_muxed_add`.

```
legacy_genus:/> report power -rtl_cross_reference
============================================================

  ...
  Technology library:     xxx yy
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
============================================================

                          Leakage   Dynamic    Total
       Design            Power(nW) Power(nW) Power(nW)
-------------------------------------------------------
mult_bit_muxed_add          71.146  1683.918  1755.064

                          Leakage   Dynamic    Total
        File         Row Power(nW) Power(nW) Power(nW)
-------------------------------------------------------
mult_bit_muxed_add.v  8     35.573   715.090   750.663
mult_bit_muxed_add.v  9     35.573   643.885   679.457
```

- The following command shows RTL power analysis for all levels of the hierarchy.

```
legacy_genus:/> report power -rtl -flat
============================================================

  ...
  Technology library:     xxx yy
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
============================================================

                          Leakage   Dynamic    Total
       Design            Power(nW) Power(nW) Power(nW)
-------------------------------------------------------
mult_bit_muxed_add          71.146  1683.918  1755.064

                          Leakage   Dynamic    Total
        File         Row Power(nW) Power(nW) Power(nW)
-------------------------------------------------------
muxed_add.v           8     14.199   551.785   565.984
muxed_add.v           9     28.473   396.138   424.611
muxed_add.v           12    28.473   411.051   439.524
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command requests a detailed RTL power analysis report.

```
legacy_genus:/> report power -rtl -detail -flat
============================================================
...
  Module:                 mult_bit_muxed_add
...
============================================================

                                                   Leakage   Internal    Net
       Design                                     Power(nW) Power(nW) Power(nW)
-------------------------------------------------------------------------------
mult_bit_muxed_add                                   71.146  1148.102   535.815
-------------------------------------------------------------------------------

                                                   Leakage   Internal    Net
    Primary Input                                 Power(nW) Power(nW) Power(nW)
-------------------------------------------------------------------------------
a[1]                                                  0.000     0.000    34.050
a[0]                                                  0.000     0.000    34.050
b[1]                                                  0.000     0.000    34.050
b[0]                                                  0.000     0.000    34.050
c[1]                                                  0.000     0.000    34.050
c[0]                                                  0.000     0.000    34.050
d[1]                                                  0.000     0.000    34.050
d[0]                                                  0.000     0.000    34.050
s                                                     0.000     0.000    52.536
--------------------------------------------------------------------------------

                                                   Leakage   Internal    Net
        File         Row        RTL Line Instances Power(nW) Power(nW) Power(nW)
-------------------------------------------------------------------------------
muxed_add.v          8  {if (s) begin} mux..8_5_g1    14.199   469.091   82.695
                                       mux_y_8_5_g1
-------------------------------------------------------------------------------
muxed_add.v          9   {y = a + c;}           g2    28.473   334.117    62.021
                                                g2
-------------------------------------------------------------------------------
muxed_add.v          12  {y = b + d;}           g2    28.473   344.894    66.156
                                                g2
-------------------------------------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command shows the clock tree power estimation for clock `clk`.

```
legacy_genus:/> report power -clock_tree iCLK1 -width 5 -height 5
============================================================
  Generated by:           Genus Synthesis Solution version
  Generated on:           date
  Module:                 test
  Technology libraries:   typical 1.3
  Operating conditions:   typical (balanced_tree)
  Wireload mode:          segmented
  Area mode:              timing library
============================================================

Clock Power Estimation Summary for clock 'iCLK1'
================================================

-----------------------------------------------------
Estimate   Leakage (nW)   Dynamic (nW)   Total (nW)
-----------------------------------------------------
Max               0.007     147560.871     147560.878
Min               0.007      42160.249      42160.251
Typical           0.007      67941.241      67941.244

Leaf CGIC Cells                        4
Leaf Clock Buffers                     0
Total Clock Buffers                    2

Estimation Parameters
=====================

Clock Buffers Used: BUFX12 BUFX16 BUFX2
                    BUFX20 BUFX3 BUFX4
                    BUFX6 BUFX8 CLKBUFX12
                    CLKBUFX16 CLKBUFX2 CLKBUFX20
                    CLKBUFX3 CLKBUFX4 CLKBUFX6
                    CLKBUFX8 DLY1X1 DLY1X4
                    DLY2X1 DLY2X4 DLY3X1
                    DLY3X4 DLY4X1 DLY4X4

Max flops driven by one leaf buffer: 3
Die width: 5.0 um
Die height: 5.0 um
```

- The following command reports the power (after synthesis) at the current level in the hierarchy, which is the design.With the `-hier` option specified, the report lists the design and its hierarchical instances.

```
legacy_genus:/designs/mult_bit_muxed_add> report power -hier
============================================================
  ...
============================================================

                          Leakage   Dynamic    Total
     Instance      Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------------
mult_bit_muxed_add     6    71.146  1683.862  1755.008
  ma0                  3    35.573   724.869   760.442
  ma1                  3    35.573   643.092   678.665
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports the power (after synthesis) at the current level in the hierarchy, which is the design. With the `-verbose` option specified, the report shows in addition the components of the dynamic power.

```
legacy_genus:/designs/mult_bit_muxed_add> report power -verbose
============================================================
  ...
============================================================
                        Leakage   Internal    Net      Dynamic        Total
                                                      (Int+Net)     (Leak+Dyn)
     Instance     Cells Power(nW) Power(nW) Power(nW) Power(nW)      Power(nW)
-----------------------------------------------------------------------------
mult_bit_muxed_add    6    71.146  1148.047   535.814  1683.862      1755.008
  ma0                 3    35.573   608.229   116.640   724.869       760.442
  ma1                 3    35.573   539.817   103.275   643.092       678.665
```

- The following command reports the power (after synthesis) at the current level in the hierarchy, which is the design.With the `-flat` option specified, the report lists leaf instances starting from the current position in the hierarchy.

```
legacy_genus:/designs/mult_bit_muxed_add> report power -flat
============================================================
  ...
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
ma0/g38           13.900   266.298   280.198
ma0/g39           13.900   263.897   277.797
ma1/g38           13.900   233.010   246.910
ma1/g39           13.900   263.877   277.777
ma0/g37            7.774   194.673   202.447
ma1/g37            7.774   146.204   153.978
```

- The following command reports the power (after synthesis) at the current level in the hierarchy, which is a subdesign.With the `-hier` option specified, the report lists the subdesign and its hierarchical instances. Because in this case there are no hierarchical instances, only the power for the subdesign is listed.

```
legacy_genus:/designs/mult_bit_muxed_add/instances_hier/ma0> report power
-hier
============================================================
  ...
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
ma0          3    35.573   724.869   760.442
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports the power for an instance and sorts the report according to descending net power.

```
legacy_genus:/designs/mult_bit_muxed_add> report power -flat instances_hier/
ma0 \
-sort net
============================================================
  ...
============================================================

               Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
ma0/g39           13.900   266.298   280.198
ma0/g38           13.900   263.897   277.797
ma0/g37            7.774   194.673   202.447
```

- The following command reports the power for an instance and a net.

```
legacy_genus:/designs/mult_bit_muxed_add> report power nets/a[1] ma0 -flat
============================================================
  ...
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
ma0/g39           13.900   266.298   280.198
ma0/g38           13.900   263.897   277.797
ma0/g37            7.774   194.673   202.447

     Net         Net            Toggle
(asserted *) Power (nW) Prob. Rate (/ns) Cap. (nF)
---------------------------------------------------
a[1]             27.945 0.500      0.021     2.300
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following design has five power domains and three power modes. The following commands show the power report for two of the modes after mapping. In this case the report has an additional column Domain (voltage) which shows for each instance to which power domain it belongs and what the voltage is of the domain in the reported mode. Note that when the second `report power` command is given, the tool adjusts the wireload models for the specified mode before reporting power.

```
legacy_genus:/designs/counter> report power -power_mode PMdefault
============================================================
 ...
 Module:          counter
 Library domain:     umc_0p8v
  Domain index:     0
  Technology libraries: ...
  Operating conditions: _nominal_ (balanced_tree)
 Library domain:     umc_1v
  Domain index:     1
  ...)
 Library domain:     umc_1p2v
  Domain index:     2
  ...
 Wireload mode:     enclosed
 Area mode:       timing library
 Power mode:       PMdefault
============================================================

                     Domain           Leakage   Dynamic    Total
    Instance       (Voltage)   Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------------------------
counter          PDcore(0.81v)   142     7.076  9192.022  9199.099
  monitor_power   PDmon(0.81v)    84     3.720  4050.236  4053.956
  adder_counter   PDadd(0.81v)    25     1.215  1573.961  1575.175
  bcd_counter     PDbcd(0.81v)    12     1.088  1697.572  1698.659
  binary_counter  PDbin(0.81v)    18     1.012  1565.877  1566.888
legacy_genus:/designs/counter> report power -power_mode PMmid
============================================================
 ...
 Power mode:       PMmid
============================================================


    Applying wireload models.
Info  : Changing wireload model of a design/subdesign. [TIM-92]
    : Changing wireload model of design 'counter' from <none> to cmos065.
    : The change of wireload model will likely change the design's timing
slightly.

...
    Computing net loads.

                  Library        Leakage   Dynamic    Total
    Instance      Domain Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------------------
counter           umc_1v   142    11.023 33056.713 33067.735
  monitor_power   umc_1v    84     5.805 14774.642 14780.447
  adder_counter   umc_1v    25     1.884  4185.202  4187.086
  bcd_counter     umc_1v    12     1.697  6138.779  6140.477
  binary_counter  umc_1v    18     1.569  6542.145  6543.714
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands:	syn_generic on page 526
	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	cell_leakage_power
	leakage_power_scale_in_nW
	lp_asserted_probability
	lp_asserted_toggle_rate
	lp_power_unit
Related attributes	lp_clock_tree_buffers
	lp_clock_tree_leaf_max_fanout
	lp_computed_probability
	lp_computed_toggle_rate
	lp_leakage_power

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power_intent

```
report power_intent
     [-design design]
     [-isolation_rule_only | -level_shifter_rule_only
     |-power_domain_only | -state_retention_rule_only]
     [-domain domain] [-show_full_names]
     [-power_states] [-supply string[-summary] [-detail] [> file]
```

Prints out the power intent included in the power intent file that was read in.

Options and Arguments

`-design` *design*	Specifies the design for which to create the report. If no design is specified, the report is given for the current design. This option is required when multiple designs are loaded.
`-detail`	Requests a detailed report.
`-domain` *domain*	Reports all the rules defined for the specified domain.
file	Specifies the name of the file to which to write the report.
`-isolation_rule_only`	Reports the isolation rules for the power domain they were defined.
`-level_shifter_only`	Reports the level shifter rules for the power domain they were defined.
`-power_domain_only`	Reports only the power domains.
`-power_states`	Prints for each power state a power state table.
`-show_full_names`	Specifies to print full names in the report.
`-state_retention_rule_only`	Reports the state retention rules for the power domain they were defined.
`-summary`	Reports the number of power domains, level shifter rules, isolation rules, state retention rules, and power modes.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-supply` *`string`* Prints the power state table for the specified supply. You can specify a supply set, a power pin, port, or net, or a ground pin, port, or net.

Examples

- The following command lists the power domains in the power intent file.

```
legacy_genus:/> report power_intent -power_domain_only
======================================================================
                     Low Power Intent
                     ----------------
Type : IEEE-1801     File : test_11.upf
======================================================================

      Power Domains
      =============
          SW
          AON
--------------------------------------------
```

- The following command lists the isolation rules.

```
legacy_genus:/> report power_intent -isolation_rule_only
======================================================================
                     Low Power Intent
                     ----------------
Type : IEEE-1801     File : test_11.upf
======================================================================

      Power Domain : SW
      ==================
          Isolation Rules
          --------------
            SW.iso
            SW.iso2
            SW.iso3
            ...
            SW.iso8
            SW.iso9
```

- The following command lists the rules for domain SW.

```
legacy_genus:/> report power_intent -domain SW
======================================================================
                     Low Power Intent
                     ----------------
Type : IEEE-1801     File : test_11.upf
======================================================================

      Power Domain : SW
      ==================
          Isolation Rules
          --------------
            SW.iso
            SW.iso2
            SW.iso3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
            ...
            SW.iso8
            SW.iso9
```

- The following report lists the quality issues of the power intent file.

```
legacy_genus:/> report power_intent -quality_check


        Low Power_Intent Quality Check Summary
================================
Type: IEEE-1801    test_11.upf
================================

PD : Design object has no power-domain or supply-set. ISO/
LS Rule might not be
applicable.
  Ports         :       7


Rule : Multiple rules for domain. In case of conflict one with higher precede
nce
will be applicable.
  Domain : SW
    ISO-Rule    : 9
================================
```

- The following command requests a detailed report. The report prints the details of the rules.

```
legacy_genus:/> report power_intent -det
=====================================================================
                     Low Power Intent
                     ----------------
Type : IEEE-1801     File : test_11.upf
=====================================================================

      Power Domain : SW
      ==================
          Isolation Rules
          --------------
            SW.iso
                domain : SW
                source : SW
                sink : AON
                location : self
                applies_to : inputs
                clamp_value : 0
                isolation_signal : Iso
                isolation_sense : high
                isolation_target : from
                source_off_clamp : 0
            SW.iso2
                domain : SW
                sink : AON
                location : self
                applies_to : inputs
                clamp_value : 0
                isolation_signal : Iso
                isolation_sense : high
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
                 isolation_target : to
                 sink_off_clamp : 0
                .....
```

- The following report gives the summary of the power intent file.

```
legacy_genus:/> report power_intent -summary
Summary
=====================================================
Power Intent File (format: IEEE-1801) : test_11.upf
=====================================================
Number of Power Domains               :      2
Number of Isolation Rules             :      9
Number of Level Shifter Rules         :      0
Number of State Retention Rules       :      0
Number of Power Modes                 :      2
=====================================================
```

Related Information

Affected by these commands: apply_power_intent on page 1268
commit_power_intent on page 1281
read_power_intent on page 1285

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power_intent_instances

```
report power_intent_instances [-design design]
     [-instance instance] [-lp_instance instance]
     [-isolation_only | -level_shifter_only |-combo_only
     |-state_retention_only] [-show_full_names]
     [-rule {isolation_rule | level_shifter_rule
          | state_retention_rule}]
     [-summary] [-detail] [> file]
```

Reports information about the low power cells inserted in the design.

Note: This command also reports pre-inserted low power cells.

Options and Arguments

`-combo_only`	Reports isolation-level shifter combo cells only.
`-design` *design*	Specifies the design for which to create the report. If no design is specified, the report is given for the current design. This option is required when multiple designs are loaded.
`-detail`	Requests a detailed level shifter report, including pins on which the low power cells were inserted.
file	Specifies the name of the file to which to write the report.
`-instance` *instance*	Restricts the reported information to the specified hierarchical instance.
`-isolation_only`	Reports isolation cells only
`-level_shifter_only`	Reports level shifters only.
`-lp_instance` *instance*	Reports detailed information for the specified low power cell instance.
`-rule {`*isolation_rule*|*level_shifter_rule*|*state_retention_rule*`}`	Reports the low power cells inserted for the specified rule.
`-show_full_names`	Specifies to print full names in the report.
`-state_retention_only`	Reports state retention cells only.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-summary` Reports the number of level shifters, isolation cells, combo cells, and state retention cells inserted in the design.

Examples

- The following report gives the summary of the inserted low power cells.

```
legacy_genus:/> report power_intent_instances -summary
==============================================================
  Generated by:          Genus Synthesis Solution version
...
==============================================================

Summary
========================================
Number of Isolation cells:          1305
Number of Level Shifter cells:      2213
Number of Combo (ISO+LS) cells:       0
Number of SRPG cells:                 0
========================================
```

- The following report shows the number of level shifters inserted per rule.

```
legacy_genus:/> report power_intent_instances -design top -level_shifter_only
==============================================================
  Generated by:           Genus Synthesis Solution version
...
==============================================================


================================
  Level Shifter Report
================================
  ----------------------------------------
  Design/Instance                  top
  ----------------------------------------
  Rule : pd2.ls_self_hiconn     Count : 2
  ----------------------------------------
  Total Level Shifter:      2
================================
```

- The following report shows the pin on which the level shifter was inserted, the power domains between which the cell is shifting, the name of the level shifter instance.

```
legacy_genus:/> report power_intent_instances -design top -detail
==============================================================
  Generated by:           Genus Synthesis Solution version
...
==============================================================


================================
  Level Shifter Report
================================

-------------------------------------------------------------------------------
 Design/Instance                                                     top
 -------------------------------------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
 Rule : pd2.ls_self_hiconn                                  Count : 2
   u1/in                  (pd1 -> pd1)   pd2_ls_self_hiconn_0 (LVLHLD8BWPHVT)
   u1/out                 (pd1 -> pd2)   pd2_ls_self_hiconn_1 (LVLHLD8BWPHVT)
-------------------------------------------------------------------------------
  Total Level Shifter:      2
============================================
```

- The following report shows detailed information about one level shifter instance.

```
legacy_genus:/> report power_intent_instances -design top -lp_instance
pd2_ls_self_hiconn_1
============================================================
  Generated by:           Genus Synthesis Solution version
...
============================================================

================================================================
Inst-Name                : pd2_ls_self_hiconn_1
Libcell                  : LVLHLD8BWPHVT (Direction: down)
Rule                     : pd2.ls_self_hiconn (Direction: both)
Type                     : {Level Shifter}
From_Domain(s)           : pd1
To_Domain(s)             : pd2
Location                 : parent
Function                 : LS_ONLY
Enable-pin               :
Enable-driver            :
Interface-pin            : u1/out
----------------------------------------------------------------
Driver/Load Details:
----------------------------------------------------------------
      Driver Pins(s)                  Load Pin(s)
----------------------------------------------------------------
pd2_ls_self_hiconn_0/Z                    out
================================================================
```

Related Information

Affected by these commands: apply_power_intent on page 1268
commit_power_intent on page 1281

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report proto

`report proto [-hdl] [-feasible_targets [-num integer]] [> file]`

Reports information about the prototype synthesis flow.

Important

Limited access feature.

Options and Arguments

file	Specifies the file to which the report must be written.
-feasible_targets	Prints the endpoints whose timing constraints are relaxed with path adjust exceptions. The results are sorted first according to the path adjust value, then according to the path adjust exception name, then according to the endpoint names.
-hdl	Prints a summary of incomplete RTL scenarios found during elaboration.
-num *integer*	Specifies the number of endpoints to be printed according to the decreasing path adjust value. If this option is not specified, all paths with path_adjust values are printed.

Examples

- The following command prints a list of endpoints whose timing constraints were relaxed.

```
legacy_genus:/> report  proto  -feasible_targets

==============================================================
  ...
==============================================================

=============================================================
No.  Endpoint              Path Adjust       Path Adjust
                              Value(ps)       Exception
=============================================================

1   top/piano/u_reg/d         1500         proto_ft_adj_1
2   top/piano/p_reg/d         1500         proto_ft_adj_2
3   top/piano/p_reg/d         1500         proto_ft_adj_2
4   top/q_reg1/d               450         proto_ft_zipped_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
5   top/q_reg2/d              450        proto_ft_zipped_2
============================================================
```

- The following command prints a list of incomplete RTL scenarios.

```
legacy_genus:/> report proto -hdl
============================================================
  ...
============================================================

INCOMPLETE HDL SCENARIO FOR DESIGN: /designs/top

PORTS SEEN IN INSTANCES BUT MISSING IN MODULE DEFINITION
=========================================================

PORT NAME:     in2
INSTANCE NAME: u1
FILE NAME:     missing_port.v
MODULE NAME:   mid

INCOMPLETE HDL SCENARIO FOR SUBDESIGN: /designs/top/subdesigns/bot

PORTS SEEN IN INSTANCES BUT MISSING IN MODULE DEFINITION
=========================================================

PORT NAME:     in3
INSTANCE NAME: I1
FILE NAME:     missing_port.v
MODULE NAME:   mid
```

Related Information

Affected by this command:	elaborate on page 505
	syn_map on page 529
Related attributes:	proto_feasible_target
	proto_hdl

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report qor

```
report qor
     [-levels_of_logic] [-nopower | -power]
     [-exclude_constant_nets] [-ndr_min_layer]
     [design]... [> file]
```

Reports the critical path slack, total negative slack (TNS), number of gates on the critical path, and number of violating paths for each cost group. It also gives the instance count, total area (net and cell area), cell area, runtime, and host name information.

Tip

Power information is only reported by default if you set at least one of these power constraint attributes: `max_leakage_power` or `max_dynamic_power`.

If you perform physical synthesis and start with a floorplan, the report also contains the floorplan utilization in %. If you executed the `synthesize -to_placed` command, the report will also contain a `Silicon Virtual Prototype` section that lists the total and average net length in micron, and the routing congestion in %. Routing congestion is a measure of track overflow. A value greater than 5% in either direction gives an indication that the design will be difficult to route. The information is static information from the most recent Innovus® batch job.

Note: In case of a multi-mode design, the TNS column is not printed.

Options and Arguments

design	Specifies the design name on which to report. If no design is specified, the report is given for the current design.
`-exclude_constant_nets`	Excludes constant nets from the fanout statistics computation. This can affect the `Max Fanout, Min Fanout, Average Fanout` and the `Terms to net ratio` numbers in the report.
file	Specifies the file to which the report must be written.
`-levels_of_logic`	Prints the number of gates on the critical path per cost group.
`-ndr_min_layer`	Reports the number of non-default routes and nets assigned to higher metal layers by Innovus.
`-nopower`	Prevents power computation for this report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-power` Forces to compute and report the power results when the power constraints were not set.

Examples

- The following example reports the QoR data for the current design. Since no power constraints were set, by default no power information is included in the report.

```
legacy_genus:/> report qor
============================================================
...
  Module:                  cscan
  Technology libraries:    tutorial 1.0
                           slow_hvt 1.1
  Operating conditions:    typical_case (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
============================================================

Timing
------

Clock Period
-------------
CLK1  6000.0

Cost   Critical        Violating
Group  Path Slack TNS      Paths
----------------------------------
C2C        No paths   0
C2O          2015.1   0          0
CLK1         2019.9   0          0
default    No paths   0
I2C          1707.2   0          0
I2O        No paths   0
-----------------------------------
Total                 0          0

Instance Count
--------------
Leaf Instance Count             39
Sequential Instance Count       16
Combinational Instance Count    23
Hierarchical Instance Count      0

Area
----
Cell Area                          304.240
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)    304.240
Net Area                           0.000
Total Area (Cell+Physical+Net)     304.240

Max Fanout                         16 (n_13)
Min Fanout                         1 (DFT_sdo_1)
Average Fanout                     2.2
Terms to net ratio                 2.6
Terms to instance ratio            3.6
Runtime                            11.000 seconds
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Elapsed Runtime                     19 seconds
RC peak memory usage:               121.00
EDI peak memory usage:              no_value
Hostname                            rcopt55
```

- The following example reports the QoR data for a design for which power constraints were set. In this case power information is included by default.

```
legacy_genus:/> report qor
============================================================
...
============================================================

Timing
------

Clock Period
-------------
CLK1  6000.0

Cost   Critical        Violating
Group  Path Slack TNS      Paths
----------------------------------
C2C        No paths   0
C2O          2015.1   0          0
CLK1         2019.9   0          0
default    No paths   0
I2C          1707.2   0          0
I2O        No paths   0
-----------------------------------
Total                 0          0

Instance Count
--------------
Leaf Instance Count             39
Sequential Instance Count       16
Combinational Instance Count    23
Hierarchical Instance Count      0

Area
-----
Cell Area                           304.240
Physical Cell Area                  0.000
Total Cell Area (Cell+Physical)     304.240
Net Area                            0.000
Total Area (Cell+Physical+Net)      304.240

Power
-----
Leakage Power                       179.155 nW
Dynamic Power                       677699.431 nW
Total Power                         677878.586 nW


Max Fanout                          16 (n_13)
Min Fanout                          1 (DFT_sdo_1)
Average Fanout                      2.2
Terms to net ratio                  2.6
Terms to instance ratio             3.6
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Runtime                              11.000 seconds
Elapsed Runtime                      19 seconds
RC peak memory usage:                121.00
EDI peak memory usage:               no_value
Hostname                             rcopt55
```

- The following command requests to report the number of gates on the critical path per cost group data. This adds a `No of gates on Critical Path` column the table under `Timing`. The rest of the report does not change.

```
legacy_genus:/> report qor -levels_of_logic
============================================================
...
============================================================

Timing
------
  Cost    Critical      No of gates on  Violating
 Group   Path Slack TNS  Critical Path    Paths
--------------------------------------------------
C2C        No paths   0
C2O          2015.1   0              0          0
CLK1         2019.9   0              1          0
default    No paths   0
I2C          1707.2   0              2          0
I2O        No paths   0
--------------------------------------------------
Total                 0                         0
```

- The following example reports the QoR data for a Dynamic Voltage Frequency Scaling (DVFS) design (design with multiple power modes). In this case, an additional `Mode` column is added to the table under `Timing`. The rest of the report does not change.

```
Timing
------

       Cost  Critical Path  Violating
Mode  Group      Slack        Paths
--------------------------------------
m1   default      No paths
     I2C            -202.6          1
     C2O            -116.5          1
     C2C          No paths
     I2O          No paths
m2   default        -202.6          2
     I2C          No paths
     C2O          No paths
     C2C          No paths
...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example reports the QoR data for a design with multibit instances. In this case, the (++) annotation is used in the table under `Instance Count`. The rest of the report does not change.

```
Instance Count
--------------
Leaf Instance Count                   6
Sequential Instance Count    (++)     4
Combinational Instance Count          2
Hierarchical Instance Count           0

(++) : includes multibit instances. Use report multibit_inferencing
command to get detailed report on multibits
```

- The following example reports the QoR data after you executed the `synthesize -to_placed` command.

```
legacy_genus:/> report qor
============================================================
  Generated by:           Genus Synthesis Solution version
  ...
  Interconnect mode:      placement
  Area mode:              physical library
============================================================
Timing
--------

Clock Period
--------------
vclk01  5000.0
vclk02  6000.0
....

  Cost    Critical            Violating
 Group   Path Slack    TNS      Paths
----------------------------------------
default    No paths        0
vclk1       -4944.1   -12686         38
vclk01     No paths        0
...
----------------------------------------
Total                 -12686         38

Instance Count
--------------
Leaf Instance Count             5977
Sequential Instance Count        546
Combinational Instance Count    5431
Hierarchical Instance Count       26
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Area
----
Cell Area                          1221637.592
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)    1221637.592
Net Area                           192299.489
Total Area (Cell+Physical+Net)     1413937.081

Floorplan Utilization              37.35%
Max Fanout                         540 (scan_enI)
Min Fanout                         0 (DTMF_INST/ULAW_LIN_CONV_INST/lpcm[13])
Average Fanout                     2.6
Terms to net ratio                 3.6
Terms to instance ratio            3.9
Runtime                            15.000 seconds
Elapsed Runtime                    208 seconds
RC peak memory usage:              136.00
EDI peak memory usage:             no_value
Hostname                           rcae003
```

```
Silicon Virtual Prototype
-------------------------
Total Net Length                   484867.62  um
Average Net Length                 78.00  um
Routing Congestion                 H: 0.87% V: 0.63%
```

- The following command reports the Qor data while excluding the constant nets from the fanout calculation.

```
legacy_genus:/> report qor -exclude_constant_nets

============================================================
...
============================================================
...
Area
----
Cell Area                          166582.500
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)    166582.500
Net Area                           0.000
Total Area (Cell+Physical+Net)     166582.500

Excluding constant nets from fanout calculation
Max Fanout                         6021 (reset)
Min Fanout                         0 (GEN2[1].i_xbarstage/U_mux0/mux_muxout_52_14
g1/z)
Average Fanout                     2.6
Terms to net ratio                 3.5
Terms to instance ratio            3.9
Runtime                            41.940 seconds
Elapsed Runtime                    590 seconds
RC peak memory usage:              457.00
EDI peak memory usage:             no_value
Hostname                           rcopt55
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these attributes: max_dynamic_power
max_leakage_power
report_ndr_min_layer_count

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report scan_compressibility

`report scan_compressibility -directory` *`atpg_directory`* `[> file]`

Reports the scan compressibility of a design. This command is used to report the results of the `analyze_scan_compressibility` command which determines an optimal Compression architecture for your design by performing multiple compression analysis using Modus.The compression analysis can be run on the scan mapped design prior to adding the Test compression logic.

Options and Arguments

`-directory` *`atpg_directory`*

Specifies the directory containing the ATPG compression runs created by the `analyze_scan_compressibility` command.

Default: *`current_working_directory`*`/asc`

`file`

Specifies the name of the file to which the report must be written.

Examples

- The following command reports the compressibility analysis resulting from the ATPG directory `asc0`.

```
legacy_genus:/>report scan_compressibility -directory asc0
```

```
Analyzing from work directory 'asco' .....
----------------------------------------
Results of analyze_scan_compressibility
----------------------------------------
       Design       - test
       Decompressor - broadcast
       Compressor   - misr
       mask         - wide1

Achieved compression table with fullscan topup vectors
-------------------------------------------------------
IC    TATR   TDVR    ATCov.   CL       Pat-comp  Pat-fs    Cycles        Runtime    Gatecount    Area
-----------------------------------------------------------------------------------------------------
fs    1.0    1.0     100.00%  63       -         3         336           00:00:00   -            -
2     0.3    0.5     99.91%   32       9         5         1258          00:00:01   437          24774.3999999

Achieved compression table without fullscan topup vectors
-----------------------------------------------------------
IC    TATR   TDVR    ATCov.   CL       Pat-comp  Pat-fs    Cycles
-------------------------------------------------------------
fs    1.0    1.0     100.00%  63                 3         336
2     0.4    1.2     98.76%   32       9         -         838

Total atpg runtime for exp. 00:00:01 hrs.

IC        - Inserted compression
TATR      - Test application time reduction
TDVR      - Test data volume reduction
Cov.      - Atpg coverage
CL.       - Channel Length
Pat-comp  - No. of compression test patterns
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Pat-fs    - No. of fullscan test patterns
Runtime   - Atpg runtime
fs        - fullscan run
```

Related Information

Affected by this command: analyze_scan_compressibility on page 854

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report sequential

```
report sequential
     [[-instance_hier instance] [-hier] | -deleted_seqs ]
     [subdesign | design] [> file]
```

Generates a report on the sequential elements of the current design. The report provides the sequential element name, row, column, filename information, and the sequential element type (flip-flop (async set/rest, sync set/reset, sync enable), latch, or timing model).

Note: Set the `hdl_track_filename_row_col` attribute to `true` before using the `elaborate` command to track the filename, row and column information.

Options and Arguments

file	Specifies the name of the file to which to write the report.
-deleted_seqs	Reports the sequential elements that were removed during optimization. For example, sequential elements can be removed during constant propagation, redundancy removal, when merged with other sequential instances, or when they are not driving any primary outputs. **Note:** If the `delete_unloaded_seqs` attribute is set to `false`, the sequential elements that were optimized but did not get deleted will also be reported. The reason for these flops is reported with an asterisk (*) . In this case, the flop's output pin will be dangling, not driving any loads. The loads driven by the flop before optimization, will be driven by a constant, or by the output pins of the merged flops pin. **Note:** This option cannot be combined with any other option.
-hier	Reports all the flops in the design.
-instance_hier *instance*	Specifies the hierarchical instance name to report flops.
subdesign | *design*	Specifies the design or subdesign name to report flops.

Examples

- The following example reports all the sequential elements in the top-level design.

```
legacy_genus:/> report sequential
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
============================================================
  Generated by:           Version
  Generated on:           Date
  Module:                 test
  Technology library:     slow 1.5
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          segmented
============================================================

                                Instantiated/
 Register       File  Row Column  Inferred              Type
-----------------------------------------------------------------------------
sync_rst_reg    all.v  12   16   inferred      flip-flop synchronous reset
async_rst_reg   all.v  21   27   inferred      flip-flop asynchronous reset
sync_set_reg    all.v  30   37   inferred      flip-flop asynchronous set
no_rst_reg      all.v  39   45   inferred      flip-flop
sync_preset_reg all.v  44   58   inferred      flip-flop synchronous set
q_reg           all.v  6    12   inferred      latch
```

- The following example reports all the sequential elements in the design.

```
legacy_genus:/> report sequential -hier
report sequential: prints a sequential instance report.
============================================================
....
============================================================

                                            Instantiated/
  Register                File     Row Column    Inferred          Type
-------------------------------------------------------------------------------
m2/m3/m4/m5/o_m5_0_reg_1 hier.v  25   22  instantiated  flip-flop synchronous
                                                                      enable
m2/m3/m4/m5/o_m5_0_reg_2 hier.v  27   22  instantiated  flip-flop synchronous
                                                                      enable
m2/m3/m4/m5/o_m5_1_reg_0 hier.v  30   22  instantiated  flip-flop synchronous
                                                                      enable
m2/m3/m4/m5/o_m5_0_reg_0 hier.v  23   22  instantiated  flip-flop synchronous
                                                                      enable
...................
...................
m2/o_m2_clk1_0_reg_2     hier.v 174   27  instantiated  flip-flop synchronous
                                                                      enable
```

- The following example reports a timing model.

```
legacy_genus:/> report sequential
================================================================================
...
================================================================================
                                                Instantiated/
 Register                File           Row Column    Inferred        Type
--------------------------------------------------------------------------------
clockgate/g2clatch_tlat timing_model.v  22   29   instantiated  timing_model
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example reports the sequential elements that were removed during optimization.

```
legacy_genus:/> report sequential -deleted_seqs

================================================================================
...
================================================================================

  Reason      Instance Name
------------------------------
merged         ctl_wr_reg[36] merged with ctl_wr_reg[0]
merged         ctl_wr_reg[46] merged with ctl_wr_reg[10]
merged         ctl_wr_reg[47] merged with ctl_wr_reg[11]
merged         ctl_wr_reg[40] merged with ctl_wr_reg[12]
merged         ctl_wr_reg[41] merged with ctl_wr_reg[13]
...
...
merged         ctl_wr_reg[3] merged with ctl_wr_reg[39]
merged         ctl_wr_reg[8] merged with ctl_wr_reg[44]
merged         ctl_wr_reg[9] merged with ctl_wr_reg[45]
unloaded       ctl_wr_reg[32]
unloaded       ctl_wr_reg[12]
unloaded       ctl_wr_reg[20]
unloaded       ctl_wr_reg[28]
```

If the `delete_unloaded_seqs` root attribute is set to `false`, the report may look like:

```
legacy_genus:/> report sequential -deleted_seqs

================================================================================
...
================================================================================

  Reason      Instance Name
------------------------------
merged ( * ) ctl_wr_reg[ 36 ] merged with ctl_wr_reg[ 0 ]
merged ( * ) ctl_wr_reg[ 46 ] merged with ctl_wr_reg[ 10 ]
merged ( * ) ctl_wr_reg[ 47 ] merged with ctl_wr_reg[ 11 ]
merged ( * ) ctl_wr_reg[ 40 ] merged with ctl_wr_reg[ 12 ]
merged ( * ) ctl_wr_reg[ 41 ] merged with ctl_wr_reg[ 13 ]
constant 0 ( * ) ctl_wr_reg[ 52 ]
constant 1 ( * ) ctl_wr_reg[ 53 ]


(*) indicates that the instance is optimized but not deleted
    because root attribute 'delete_unloaded_seq' is set to false
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report slew_calculation

```
report slew_calculation  pin
     [-rise | -fall]
     [-mode mode] [> file]
```

Reports how the slew of a cell driver pin is calculated from the look up table in the loaded technology library. The formula for calculating the delay is provided at the bottom of the report.

Options and Arguments

file	Redirects the report to the specified file.
[-fall | -rise]	Uses the falling or rising slew calculation on the driver pin. By default, all possible arcs are reported.
-mode *mode*	Specifies that the report should only apply to the specified *analysis_view*.
pin	Specifies the cell driver pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report summary

```
report summary
     [-mode mode] [-all]
     [[> file]
```

Reports the area by mode used by the design, cells mapped for the blocks in the specified design, the wireload model, and the timing slack of the critical path. It also reports if any design rule is violated and the worst violator information.

Options and Arguments

`-all`	Reports all timing and DRC violations in the design.
`file`	Specifies the name of the file to which to write the report.
`-mode` *`mode`*	Specifies the mode for which the report must be specified. **Note:** This option is only required for a design using a dynamic voltage frequency scaling methodology.

Example

- The following example generates a report of the area used by the design, the worst timing endpoint in the design, and the design rule violations summary.

```
legacy_genus:/> report summary
============================================================
  Generated by:          Genus Synthesis Solution version
  Generated on:          date
  Module:                alu
  Technology library:    tutorial 1.0
  Operating conditions:  typical_case (balanced_tree)
  Wireload mode:         enclosed
============================================================
        Timing
        ------
 Slack       Endpoint
-------------------------------------
 -1082ps out1_tmp_reg[9]/D
        Area
        ----
Instance       Cells   Cell Area   Net Area     Wireload
-----------------------------------------------------------
gen_test         326         525          0    AL_MEDIUM (S)
 (S) = wireload was automatically selected
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
        Design Rule Check
        -----------------
Max_transition design rule: no violations.
Max_capacitance design rule (violation total = 19402.5)
Worst violator:
Pin                          Load (ff)           Max     Violation
-----------------------------------------------------------------
in0[5] (Primary Input)           96.9           5.0          91.9
Max_fanout design rule (violation total = 16.000)
Worst violator:
Pin                             Fanout           Max     Violation
-----------------------------------------------------------------
in0[5] (Primary Input)          8.000         4.000         4.000
```

Related Information

Affected by these commands:

create_mode on page 453

syn_generic on page 526

syn_map on page 529

syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report test_power

```
report test_power
     [[-clock float][-flop_toggle_percentage float]
     | -atpg  -library string  [-clock float]
       [-directory path] [-capture | -scan_shift]
       [-lower_bound | -upper_bound]
       [-compression_mode mode]
     | -tcf file] [-view analysis_view]
     [-power_mode {mode|power_mode}] [> file]
```

Reports the estimated average power consumption of the design during test. Additionally, when using the `-atpg` option, the command reports the average switching activities for the scan shift or capture mode. The test power will be estimated using the power tables in the synthesis library (.lib).

This command should be run after the design is mapped to technology cells and scan chains are connected. Test power can only be estimated in a scan-mapped design with functioning scan chains. If the netlist inserted into the synthesis session has existing scan chains, scan chains must first be analyzed using the `define_dft scan_chain -analyze` before executing this command.

Options and Arguments

`-atpg`	Invokes Encounter Test to compute the toggle activity from the test patterns. **Note:** To use this option, you must first build the scan chains using the `connect_scan_chains` command. Or if you start from a netlist with existing scan chains, you must first analyze the scan chains using the `define_dft scan_chain -analyze` command before you can use this option.
`-capture`	Estimates the average power consumed during the capture mode of the test patterns. If neither the `-scan_shift` or `-capture` option are specified, the default is `-scan_shift`.
`-clock` *`float`*	Specifies the frequency in MHz for the specified test mode (capture or scan). *Default*: frequency of the test clock with highest frequency.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-compression_mode {COMPRESSION | COMPRESSION_DECOMP | OPMISRPLUS | OPMISRPLUS_DECOMP}`

Specifies which compression test mode to select for power analysis.

If you omit this option, the default test mode that will be used is `FULLSCAN`.

`-directory` *`string`*

Specifies the work directory to which the script files must be written. If the script files exist, the command can overwrite them.

Default: *`current_working_directory`*`/test_power`

`file`

Specifies the file to which to redirect the report.

`-flop_toggle_percentage` *`float`*

Specifies the percentage of the flops that are changing state in each test clock cycle.

Default: 50% of the activity level in the design.

`-library` *`string`*

Specifies the list of Verilog structural library files. Specify the list in a quoted string.

Note: This option is only required when you invoke this command on a mapped netlist.

`-lower_bound`

Runs ATPG with repeat fill to provide a lower bound on the estimated power.

If you specified neither the `-lower_bound` nor the `-upper_bound` option, the `-upper_bound` option will be used by default.

Note: This option must be specified with the `-atpg` option.

`-power_mode {`*`mode`*`|`*`power_mode`*`}`

Specifies the mode for which the test power must be reported. You can either specify a mode or power mode name.

Note: This option is required when the design has multiple power modes.

`-scan_shift`

Reports the average power consumed in scan shift mode.

If neither the `-scan_shift` or `-capture` option are specified, the default is `-scan_shift`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-tcf file`
: Specifies the TCF file from which to read the switching activities.

`-upper_bound`
: Runs ATPG with random fill to provide an upper bound on the estimated power.

  If you specified neither the `-lower_bound` nor the `-upper_bound` option, the `-upper_bound` option will be used by default.

  Note: This option must be specified with the `-atpg` option.

`-view analysis_view`
: Reports the power for the given analysis view.

Examples

- The following three sets of commands are equivalent. They all specify two files to be used as Verilog simulation libraries.

```
set simLibs "sim/tsmc13.v sim/tpz013g3.v"
report test_power  -atpg -library $simLibs
```

```
set rootDir sim
set verilogLibs "$rootDir/tsmc13.v $rootDir/tpz013g3.v"
report test_power -atpg -library $verilogLibs
```

```
set rootDir sim
report test_power -atpg -library "${rootDir}/tsmc13.v ${rootDir}/tpz013g3.v"
```

- The following command requests to estimate the test power consumed in scan shift mode when a test clock frequency of 20 MHz is applied, using the default flop toggle percentage of 50%.

```
legacy_genus:/> report test_power -clock 20
Computing the test power with the following toggle frequencies:
... set clocks @ 20 MHz
... set flops  @ 50%
============================================================
  Generated by:           version
  Generated on:           date
  Module:                 cpu
  Technology library:     typical 1.3
  Operating conditions:   typical (balanced_tree)
  Wireload mode:          segmented
  Area mode:              timing library
============================================================

                Leakage   Internal     Net      Dynamic       Total
                                                        (Internal+Net) (Leakage+Dynamic)
Instance Cells Power(nW) Power(nW)  Power(nW)  Power(nW)    Power(nW)
-----------------------------------------------------------------------------
cpu       1398     0.417 66742.159 120719.340 187461.499     187461.916
```

- The following command requests to compute the upper bounds of the test power in scan shift mode from the ATPG scan patterns with a test clock frequency of 20 MHZ. This command requests to use the `FULLSCAN` scan structure (default).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
report test_power -atpg -clock 20 -upper_bound -directory rc_rtp_UB \
-library $simLibs
```

- The following command requests to compute the lower bounds of the test power in scan shift mode from the ATPG scan patterns with a test clock frequency of 20 MHZ. This command requests to use the `FULLSCAN` scan structure (default).

```
report test_power -atpg -clock 20 -lower_bound -directory rc_rtp_LB \
-library $simLibs
```

- The following command requests to compute the test power in capture mode from the ATPG scan patterns with a test clock frequency of 20 MHZ.

```
report test_power -atpg -clock 20 -capture -directory rc_rtp_CAP \
-library $simLibs
```

- The following command requests to compute the test power from the specified TCF file.

```
report test_power -tcf top.tcf
```

Related Information

Affected by this constraint: define_dft test_clock on page 990

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report timing

```
report timing [-endpoints] [-summary] [-lint]
     [-full_pin_names] [-physical] [-num_paths integer]
     [-logic_levels integer] [-logic_levels_tcl_list]
     [-skip_buf] [-skip_inv]
     [-slack_limit integer] [-worst integer] [-user_derate]
     [-from {instance|external_delay|clock|port|pin}...]
     [-through {instance|port|pin}...]...
     [-to {instance|external_delay|clock|port|pin}...]
     [-paths string] [-exceptions exception...]
     [-cost_group cost_group] [-mode mode] [-encounter]
     [-timing_bin bin] [-timing_path path] [-gtd]
     [-collection] [-gui] [> file]
```

Generates a timing report of the current design. By default, the report gives a detailed view of the critical path of the current design. If the current session has multiple designs, use the `cd` command to navigate to the desired design to generate the report. You can also generate a report on possible timing constraint problems (timing lint) or view slack at endpoints. The timing report header will list the libraries and operating conditions used. This behavior can be disabled using the options `dont_report_library` and `dont_report_operating_conditions`.

Note: All values are always expressed in picoseconds. This unit cannot be changed. The system time unit is picoseconds and the load unit is femtofarads.

When Genus detects a combinational feedback loop, it inserts a buffer from the technology library as a loop breaker instance before it performs timing analysis. To add the loop breaker, Genus might first need to uniquify a hierarchical instance which can result in a change in the netlist. As the module and instance name can change, this can affect your scripts and your database search. Review the log file for the impact of loop breaker insertion.

Where applicable, special footnotes are used as shown in the table below to enhance the usability of the report. A footnote is added to the report to indicate the meaning of the markers used. The complete set is as follows:

Footnote	Meaning
`(*)`	Zero Slack Borrow Limit = Maximum amount that can be borrowed without violating timing on the lending side
`(@)`	Annotated capacitance
`(#)`	NDR net
`(a)`	Net has asynchronous load pins which are being considered ideal
`(b)`	Timing paths are broken
`(B)`	Net has high fanout (fanout >= 1000)
`(C)`	Cell belongs to congested region (applies only to physical flow)

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Footnote	Meaning
(e)	Net has extra wire length
(H)	Estimated wire delay based on physical-aware mapping using virtual buffering as needed.
(i)	Net is ideal
(m)	Attribute `cell_delay_multiplier` is modified for this library cell
(P)	Instance is preserved
(p)	Instance is preserved but may be resized
(u)	Net has unmapped pin(s)
(V)	Net with virtual buffer(s)

Options and Arguments

`-collection`
Specifies to return a collection of timing paths.

`-cost_group` *`cost_group`*
Reports only paths for the specified cost groups.

`file`
Specifies the name of the file to which to write the report.

`-encounter`
Generates a timing report similar to the timing report generated by Innovus. This report includes the equation for the slack and uses the same columns.

`-endpoints`
Reports the slack at all timing endpoints in the design instead of the detailed path report. The most critical endpoints are listed first.

`-exceptions` *`exceptions`*
Reports only paths to which one of the specified exceptions applies.

Note: This option can be combined with `-from`, `-through`, `-to`, and `-endpoints` options to further restrict the path reporting.

`-from {`*`instance`*`|` *`external_delay`*`|` *`clock`* `|` *`port`* `|` *`pin`*`}`
Specifies a Tcl list of start points for the paths. The start points can be input ports of your design, clock pins of flip-flops, clock objects, or a combination of these, instances, or input ports to which the specified external delay timing applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-full_pin_names`
: Prints the full hierarchical path of each pin in the report. You can use the pin names from the report to paste into other commands. By default, the report is formatted hierarchically with the local instance name on each line, and parent module names on a separate line above their children.

`-gtd`
: Generates a machine readable timing (MRTRF) report from Genus, in place of the usual human readable output. This is a file format used for representing timing information that can be understood by the Global Timing Debug (GTD) viewer in Genus, Innovus, and Tempus. By default the worst path is reported, but -`num_paths` can be used to include more critical paths.

`-gui`
: Allows you to create a detailed timing report in the GUI without having to use the menu commands. By using this option from the command line you can fine-tune the report using all command-line options which are not all available in the dialogs.This option has only effect when you run the tool in GUI mode. The path is also highlighted in the physical view.

`-lint`
: Reports, in an abbreviated output, possible timing problems in the design. These problems can be caused by generated clocks, paths constrained with different clocks, ports that have no external delays, primary inputs that have no external driver or input transition set, primary outputs without external load, timing exceptions that cannot be satisfied, constraints that may have no impact on the design, and so on.

`-logic_levels` *`integer`*
: Prints the start and end points and logic depths of the deepest paths. The integer argument specifies the number of paths to print.

`-logic_levels_tcl_list`
: Returns the start and end points, and the number of levels of logic for the number of top paths reported in Tcl format.

`-mode` *`mode`*
: Reports the worst timing across all modes or analyzes timing in one particular mode.

`-num_paths` *`integer`*
: Specifies the maximum number of paths to report.

Default: the value of `-worst`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: When combined with the `-endpoints` option, the number of endpoints is limited to the specified number.

`-paths` *`string`*
Reports on the paths matching a path specification created by the specify_paths command.

`-physical`
Reports physical information, like the x, y location. When the `report_timing show_wire_length` root attribute is enabled, the wire length information will be shown as well.

`-skip_buf`
Excludes buffers from the logic level count.

`-skip_inv`
Excludes inverters from the logic level count.

`-slack_limit` *`integer`*
Reports only paths with a slack smaller than the specified number.

`-summary`
Generates a short timing report that includes timing slack, start-point and end-point but does not include the full path.

`-through {`*`instance`* `|` *`port`* `|` *`pin`*`}`
Specifies a set of timing points as a Tcl list. The path must pass through at least one of these points. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances. You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-timing_bin` *`bin`*
Reports the timing for all paths in the specified timing bin.

The timing bin must have been previously created with the `create_timing_bin` command.

This option can only be used in conjunction with the `-num_paths` and `-gui_phys` options.

`-timing_path` *`path`*
Reports the timing for the specified timing path.

This timing path must be part of a timing bin.

`-to {`*`instance`* `|` *`external_delay`* `|` *`clock`* `|` *`port`* `|` *`pin`*`}`
Specifies a Tcl list of endpoints for the paths. The endpoints can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which the specified external delay timing exception applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Only paths that end at one of the ports or pins, or paths that are captured by one of the clock objects have the exception applied to them.
`-user_derate`	Controls the display of an additional column for the user derating values in the timing report.
`-worst` *`integer`*	Specifies the maximum number of paths to report to each endpoint. *Default*: `1`

Examples

- The following example reports the path from input port `accum[1]` that has the least slack.

```
legacy_genus:/> report timing -from [find / -port accum[1]]
============================================================
  ...
============================================================

    Pin                 Type      Fanout  Load  Slew   Delay   Arrival
                                          (fF)  (ps)    (ps)    (ps)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clock)           capture                                3500 R
                        adjustments                     -500   3000
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :      30ps
Start-point  : accum[1]
End-point    : aluout_reg[7]/D
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example generates a report for the worst path to each of the four most-constrained endpoints. The extraction of the report shows four different endpoints.

```
legacy_genus:/designs/alu> report timing -num_paths 4
============================================================
  ...
============================================================

path   1:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clock)      capture                                 3500 R
                   adjustments                    -500     3000
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :     30ps
Start-point  : accum[1]
End-point    : aluout_reg[7]/D

path   2:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :    289ps
Start-point  : accum[1]
End-point    : aluout_reg[6]/D

path   3:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :    292ps
Start-point  : accum[1]
End-point    : aluout_reg[5]/D

path   4:

-----------------------------------------------------------------
...
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :    536ps
Start-point  : accum[1]
End-point    : aluout_reg[4]/D
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example also generates a report for the four worst paths in the current design, but compared to the previous example, all worst paths now have the same endpoint.

```
legacy_genus:/designs/alu> report timing -num_paths 4 -worst 4
============================================================
  ...
============================================================
 path   1:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------

Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :      30ps
Start-point  : accum[1]
End-point    : aluout_reg[7]/D

path   2:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------

Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :      30ps
Start-point  : data[1]
End-point    : aluout_reg[7]/D

path   3:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------

Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :      30ps
Start-point  : accum[1]
End-point    : aluout_reg[7]/D

path   4:

      Pin            Type       Fanout Load Slew Delay Arrival
                                       (fF) (ps)  (ps)   (ps)
-----------------------------------------------------------------
...
-----------------------------------------------------------------
Exception    : 'path_adjusts/adj_1' path adjust    -500ps
Cost Group   : 'I2C_cg' (path_group 'I2C_pg')
Timing slack :      30ps
Start-point  : data[1]
End-point    : aluout_reg[7]/D
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command reports the path with the least slack that uses a `multi_cycle` exception named `mc_2`.

```
legacy_genus:/> report timing -exceptions [find / -exception mc_2]
```

- The following example reports only a condensed version of the timing report.

```
legacy_genus:/> report timing -summary
============================================================
  Generated by:          Genus Synthesis Solution version
  ....
============================================================
Timing slack :      30ps
Start-point  : accum[1]
End-point    : aluout_reg[7]/D
```

- The following example illustrate how to use the `-gtd` option.

```
legacy_genus:/> report_timing -from ..... -to ..... -gtd > viol.mtarpt
```

Open `viol.mtarpt` in the global timing debug viewer in Innovus.

- The following example shows the timing report when the `-user_derate` option is specified. In this example, the `cell_delay_multiplier` of cell `ffopd` was modified to `3.0`.

```
legacy_genus:/> report timing -user_derate
Warning : Possible timing problems have been detected in this design. [TIM-11]
: The design is 'top'.
============================================================
  ....
============================================================
   Pin          Type         Fanout Load Slew  User  Delay Arrival
                                     (fF) (ps) Derate  (ps)   (ps)
-------------------------------------------------------------------
(clock clk)   launch                                          0 R
inst1/CK                                   0                  0 R
inst1/Q       fflopd     (m)      1 20.4   16  3.000  +443     443 F
inst2/A                                                  +0     443
inst2/Y       inv1               1 25.4   17  1.000   +85     528 R
inst6/A                                                  +0     528
inst6/Y       buf1               1 20.4   16  1.000  +134     662 R
inst3/D       fflopd_ckn                      1.000    +0     662
inst3/CKN     setup                       0          +100     762 F
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)   capture                                        50 F
-------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :    -712ps (TIMING VIOLATION)
Start-point  : inst1/CK
End-point    : inst3/D

(m) : Attribute cell_delay_multiplier is modified for this library cell.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows the timing report when when you use the `-lint` option. In this example 5 possible timing problems were found.

```
legacy_genus:/> report timing -lint
============================================================

...
============================================================
-----------------------------------------------------------------------------
Sequential clock pins with multiple clock waveforms

The following sequential clock pins have multiple clock waveforms from a single
timing mode driving them.  ......

/designs/test/instances_hier/sub/instances_seq/f0/pins_in/CK
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Inputs without external driver/transition

The following primary inputs have no external driver or input transition set.
As a result the transition on the ports will be assumed as zero. .....

/designs/test/ports_in/in[0]
/designs/test/ports_in/in[1]
/designs/test/ports_in/in[2]
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Outputs without external load

The following primary outputs have no external load set. As a result the load
on the ports will be assumed as zero. .....

/designs/test/ports_out/out
-----------------------------------------------------------------------------


Lint summary
 Unconnected/logic driven clocks                                  0
 Sequential data pins driven by a clock signal                    0
 Sequential clock pins without clock waveform                     0
 Sequential clock pins with multiple clock waveforms              1
 Generated clocks without clock waveform                          0
 Generated clocks with incompatible options                       0
 Generated clocks with multi-master clock                         0
 Paths constrained with different clocks                          0
 Loop-breaking cells for combinational feedback                   0
 Nets with multiple drivers                                       0
 Timing exceptions with no effect                                 0
 Suspicious multi_cycle exceptions                                0
 Pins/ports with conflicting case constants                       0
 Inputs without clocked external delays                           0
 Outputs without clocked external delays                          0
 Inputs without external driver/transition                        3
 Outputs without external load                                    1
 Exceptions with invalid timing start-/endpoints                  0

                                                   Total:         5
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example reports the physical information (if the design has been placed):

```
legacy_genus:/> report timing -physical
============================================================
  ....
============================================================
      Pin                Type     Fanout Load Slew Delay Arrival    Location
                                         (fF) (ps)  (ps)   (ps)       (x,      y)
-------------------------------------------------------------------------
(clock clock1)         launch                                  0 R
wr_addr_reg[0]/CK                                0             0 R
wr_addr_reg[0]/Q  (@)  SDFFRHQX1      2  6.0  118  +311     311 R  (107640, 51660)
g154/A                                         118    +0     311
g154/Y            (@)  CLKMX2X2       3 10.9  105  +206     517 R  (102580, 51660)
g146/B                                         105    +0     517
g146/CO           (@)  ADDHXL         1  4.6  167  +189     706 R  (101660, 59040)
g145/A                                         167    +0     706
...
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clock1)         capture                             10000 R
-------------------------------------------------------------------------
Timing slack :    7997ps
Start-point  : wr_addr_reg[0]/CK
End-point    : wr_addr_reg[7]/D

(@) : Annotated capacitance.
```

- The following report shows the timing report with the **-encounter** option.

```
legacy_genus:/> report timing -encounter
============================================================
  ....
============================================================
Path 1: MET Setup Check with Pin ff2/CK
Endpoint:   ff2/D (v) checked with leading edge of 'CK'
Beginpoint: ff1/Q (^) triggered by leading edge of 'CK'
Other End Arrival Time          12000
- Setup                           304
- Uncertainty                    1000
= Required Time                 10696
- Arrival Time                   2373
= Slack Time                     8323
    Clock Rise Edge                         0
    + Clock Network Latency (Ideal)      2000
    = Beginpoint Arrival Time            2000
-------------------------------------------------------------------------
         Pin      Edge   Cell    Fanout Load Slew Delay  Arrival Required
                                        (fF) (ps)  (ps) Time(ps) Time(ps)
-------------------------------------------------------------------------
       ff1/CK     ^                              0          2000    10323
       ff1/Q      ^     DFFHQX1       1  7.1  160  +310     2310    10633
       b2/A       ^                                +0     2310    10633
       b2/Y       v     INVX1         1  5.7   88   +63     2373    10696
       ff2/D  <<< v     DFFHQX1                      +0     2373    10696
-------------------------------------------------------------------------
Cost Group   : 'CK' (path_group 'CK')
```

- The following command reports only the worst path being launched by clock clk1.

```
legacy_genus:/> report timing -paths [specify_paths -from clk1]
```

- The following example reports the slack at the four most-constrained endpoints.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/designs/alu> report timing -endpoints -num_paths 4
============================================================
...
============================================================
Slack       Endpoint        Cost Group
---------------------------------------
+30ps        aluout_reg[7]/D   I2C_cg
+289ps       aluout_reg[6]/D   I2C_cg
+292ps       aluout_reg[5]/D   I2C_cg
+536ps       aluout_reg[4]/D   I2C_cg
```

- The following example reports a specific path in the sub-bin `1ns_slack_2ns` in parent bin `worst1000_c2.`

```
legacy_genus:/> report timing -timing_path worst1000_c2c/1ns_slack_2ns/p_27
```

- The following example illustrates how to report and analyze a parent bin using linked physical and timing viewers.

```
legacy_genus:/> report timing -timing_bin worst1000_c2c -num_paths 100
-gui_phys
```

  Parent bin `worst1000_c2c` will be raised in a timing viewer. A physical viewer will also be raised and will be linked to the timing_gui.

- The following command reports the path contained in sub bin `1ns_slack_2ns` of parent bin `worst1000_2c`.

```
legacy_genus:/> report timing -timing_path worst1000_c2c/1ns_slack_2ns/p_27
```

- The following command shows how the same timing view can be linked to a specific physical gui.

```
legacy_genus:/> report timing -timing_path worst1000_c2c/1ns_slack_2ns/p_27
-gui -id 2
```

Related Information

Affected by these commands:

create_mode on page 453
specify_paths on page 494
syn_generic on page 526
syn_map on page 529
syn_opt on page 531
path_adjust on page 475
path_group on page 485

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	set_timing_derate on page 1638
Affected by this attribute:	cell_delay_multiplier
	dont_report_library
	dont_report_operating_conditions

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report units

```
report units
     [object] [ > file]
```

Reports the units used in the libraries.

You must have loaded the libraries before you can use this command.

Options and Arguments

object	Specifies the library for which you want to generate a report.
file	Redirects the report to the specified file.

Example

- In the following example two libraries, `slow` and `tutorial`, were loaded.

```
legacy_genus:/> report units
Units
----------------------------------------
LIBRARY                 slow
Time_unit               :1000ps
Capacitive_load_unit    :1000.0fF
Resistance_unit         :1kohm
Voltage_unit            :1V
Current_unit            :1uA
Power_unit              :nW
LIBRARY                 tutorial
Time_unit               :1000ps
Capacitive_load_unit    :1000.0fF
Resistance_unit         :1kohm
Voltage_unit            :1V
Current_unit            :1mA
Power_unit              :nW
```

Related Information

Affects these commands:	all report commands
Affected by this attribute:	lp_power_unit

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report utilization

`report utilization [>file]`

Reports the floorplan utilization for the design.

Options and Arguments

file	Redirects the report to the specified file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report yield

```
report yield [-depth integer] [-min_count integer]
     [> file]
```

Reports the yield cost and yield percentage for each instance. This command is used in the design for manufacturing (DFM) flow.

Options and Arguments

`-depth integer`	Specifies the number of levels of recursion.
`-min_count integer`	Specifies the minimum instance count per block.
file	Redirects the report to the specified file.

Example

- The following example shows the defect-limited yield impact for library cell defects.

```
legacy_genus:/> report yield

Instance     Cells   Cell Area            Cost     Yield %
-------------------------------------------------------
cpu            470         659    1.600606e-05     99.9984
  alu1         248         283    7.606708e-06     99.9992
  pcount1       65          92    2.215669e-06     99.9998
  ireg1         33          88    1.629471e-06     99.9998
  accum1        33          88    1.629471e-06     99.9998
  decode1       50          67    1.568901e-06     99.9998
```

Related Information

Affected by this command:	read_dfm on page 332
Related command:	report gates -yield
Affected by this attribute:	optimize_yield

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_longest_wires

```
report_chains_longest_wires
     [-num_wires integer]
     [-section_type {floating|ordered|both}] [-gui]
```

Reports the longest scan path connections across all the scan chains. Distances are reported in microns.

This command can be used on a placed design that has connected scan chains.

Options and Arguments

`-gui`	Highlights the airlines of the reported connections in the GUI.
`-num_wires` *`integer`*	Specifies the number of long wires to report. *Default*: `10`
`-section_type {floating|ordered|both}`	Reports wires that are reorderable (floating) or fixed (ordered) *Default*: `both`

Example

- The following example reports ten wires with a total distance of 5523.060.

```
legacy_genus:/> report_chains_longest_wires -gui
Listing 10 longest scan wires within both floating and ordered sections

Distance   Location        Chain       Segment     Scan

657.900  {300.96 35.28}  AutoChain_1  AutoChain_1  MAR_REG_STORAGE_Q_N52_reg[12]
         {485.1 509.04}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[932]
611.220  {308.22 120.96} AutoChain_1  AutoChain_1  C_REG_STORAGE_Q_N57_reg[7]
         {69.96 493.92}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[828]
587.220  {510.18 524.16} AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[865]
         {300.96 146.16} AutoChain_1  AutoChain_1  A_REG_STORAGE_Q_N46_reg[18]
569.340  {467.94 504.0}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[0]
         {402.6 0.0}     AutoChain_1  AutoChain_1  INSTR_REG_STORAGE_Q_N38_reg[26]
566.520  {522.72 25.2}   AutoChain_5  AutoChain_5  DFT_lockup_latch_neg_edge185345
         {339.24 408.24} AutoChain_5  AutoChain_5  THE_REG_FILE_REG.REGISTER_FILE_reg[59]
566.100  {529.32 277.2}  AutoChain_5  AutoChain_5  THE_REG_FILE_REG.REGISTER_FILE_reg[300]
         {8.58 322.56}   AutoChain_5  AutoChain_5  THE_REG_FILE_REG.REGISTER_FILE_reg[328]
511.140  {42.9 15.12}    AutoChain_1  AutoChain_1  PC_REG_STORED_VALUE_reg[8]
         {518.76 50.4}   AutoChain_1  AutoChain_1  DFT_lockup_latch_neg_edge
490.920  {189.42 478.8}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[710]
         {428.34 226.8}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[268]
489.300  {22.44 322.56}  AutoChain_5  AutoChain_5  THE_REG_FILE_REG.REGISTER_FILE_reg[463]
         {461.34 372.96} AutoChain_5  AutoChain_5  THE_REG_FILE_REG.REGISTER_FILE_reg[515]
473.400  {518.76 50.4}   AutoChain_1  AutoChain_1  DFT_lockup_latch_neg_edge
         {498.96 504.0}  AutoChain_1  AutoChain_1  THE_REG_FILE_REG.REGISTER_FILE_reg[961]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by this command: create_chains_database on page 911

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_total_wirelength

```
report_chains_total_wirelength
     [-sort {floating|ordered|chains|total}] [-gui]
```

Divides the wirelength for each scan chain into two categories:

- The *ordered* wirelength includes wires between components that are not reorderable.
- The *floating* wirelength lists the wirelength between components that are reorderable.

Distances are reported in microns.

This command can be used on a placed design that has connected scan chains.

Options and Arguments

`-gui` Highlights the chains in the GUI.

`-sort {floating|ordered|chains|total}`
Specifies which column to use for the sorting.
Default: `total`

Example

- The following example shows the report sorted by chain names.

```
legacy_genus:/> report_chains_total_wirelength -sort chains
Chain                          Floating Length   Ordered Length   Total Length

AutoChain_1:                       9188.280        0.000           9188.280
AutoChain_5:                       7423.620        0.000           7423.620
AutoChain_8:                       6788.400        0.000           6788.400
AutoChain_3:                       6695.220        0.000           6695.220
AutoChain_2:                       6198.660        0.000           6198.660
AutoChain_4:                       5983.020        0.000           5983.020
AutoChain_7:                       5824.440        0.000           5824.440
AutoChain_6:                       5473.200        0.000           5473.200
Total:                            53574.840        0.000          53574.840
```

Related Information

Affected by this command: create_chains_database on page 911

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related commands: report_chains_longest_wires on page 724
report_chains_wires on page 728

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_wires

```
report_chains_wires
     [-chain actual_scan_chain]
     [-color string] [-scandef_chain_name name]
```

Lists the Manhattan distance between every scan element in the specified chain. In addition, the chain can be highlighted in the GUI using airlines between the instances. Distances are reported in microns.

This command can be used on a placed design that has connected scan chains.

Options and Arguments

`-chain` *`actual_scan_chain`*
: Specifies the scan chain for which to report the wire-lengths.

`-color` *`string`*
: Specifies the color to use to highlight the specified chain.

`-scandef_chain_name` *`name`*
: Specifies the chain name in the scanDEF file.

Example

The following command shows the details for chain `AutoChain_1`.

```
report_chains_wires -chain AutoChain_1

Wirelength for chain: AutoChain_1

SI pin: DFT_sdi_1 Location: no_value

Dist To   Type      Parent         Location        Element
0.000    floating  AutoChain_1     no_value        COMPACTOR/msi[7]
0.000    floating  AutoChain_1     no_value        COMPACTOR/mso[7]
0.000    floating  AutoChain_1    {360.36 161.28}  DFT_lockup_latch_neg_edge185497
86.400   floating  AutoChain_1    {344.52 90.72}   PC_REG_STORED_VALUE_reg[23]
281.940  floating  AutoChain_1    {143.22 10.08}   PC_REG_STORED_VALUE_reg[18]
19.320   floating  AutoChain_1    {152.46 0.0}     PC_REG_STORED_VALUE_reg[22]
29.040   floating  AutoChain_1    {123.42 0.0}     PC_REG_STORED_VALUE_reg[29]
24.420   floating  AutoChain_1    {99.0 0.0}       PC_REG_STORED_VALUE_reg[25]
26.340   floating  AutoChain_1    {87.78 15.12}    PC_REG_STORED_VALUE_reg[15]
6.360    floating  AutoChain_1    {89.1 20.16}     PC_REG_STORED_VALUE_reg[14]
....
....
5.040    floating  AutoChain_1    {146.52 372.96}
THE_REG_FILE_REG.REGISTER_FILE_reg[894]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
12.060   floating  AutoChain_1    {148.5 362.88}
THE_REG_FILE_REG.REGISTER_FILE_reg[985]
146.340  floating  AutoChain_1    {249.48 317.52}
THE_REG_FILE_REG.REGISTER_FILE_reg[54]
0.000              Scan Out Pin    no_value        DFT_sdo_1

Total floating: 9188.280Total ordered: 0.000
Total: 9188.280
```

The report lists the scan elements in order from the chain's scan data input to the scan data output. The Manhattan distance is listed from the previous element to the current element. Each connection is categorized as floating or ordered and the location of the scan element is also given.

Note: The location used for each scan element is the lower left corner of that instance. However, when performing physical-based scan connection, the locations of the pins are used to determine the order of the scan elements.

Related Information

Affected by this command:	create_chains_database on page 911
Related commands:	report_chains_longest_wires on page 724
	report_chains_total_wirelength on page 726

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_cwd_check

```
report_cwd_check
     [-checkpoint string] [-check_name string]
     [-max_width string] [> file]
```

Reports the registered checking rules. With each checking rule, it lists the:

- Name of the checking rule
- Checkpoint(s) the rule is associated with
- Check proc(s) attached to the associated checkpoint(s)
- Effort level of the rule
- Severity level of the rule
- Description string of the rule

Note: The `-check_name` and `-checkpoint` options cannot be both used simultaneously.

Options and Arguments

`-check_name` *`string`*	Specifies, by name, a set of checking rules to report.
`-checkpoint` *`string`*	This switch specifies a set of checkpoints to report.
`-max_width` *`string`*	Limits the width of the columns in the table produced by this command. Limiting the width of a column to zero means removing that column from the table. This option takes a Tcl list of Tcl lists. Each sub-list represents a column in the table produced by this command. Each sub-list should have two elements: the first being name of the column (as seen in the report) and the second being an integer representing the maximum number of characters allowed for this column in the table.
`file`	Specifies the filename to store the output of the command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_dft_abstract_models

```
report_dft_abstract_models
     [-ctl] [design] file
```

Reports the scan abstract models read in using the `read_dft_abstract_models` command. The scan abstract model defines the scan chain architecture of the subdesign and is used as scan chain segments in the configuration of the top-level scan chains of the current design.

Options and Arguments

`-ctl`	Reports only scan abstract models that were read in from a Core Test Language (CTL) format (IEEE format P1450.6) file.
design	Specifies the design from which to report the abstract segment modules.
file	Specifies the name of the file to which to write the report.

Related Information

Related commands: read_dft_abstract_model on page 1113
write_dft_abstract_model on page 1162

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_dft_trace_back

```
report_dft_trace_back
     [-mode integer] [-polarity]
     [-continue] [-print] [-list]
     {pin|subport|port}
```

Returns the pin or port found by tracing back one level from the specified pin or port based on the requested mode. If a constant is encountered, the command returns 0 or 1.

This command uses the default DFT mode to perform constant propagation, but alternative modes can be selected by running `check_dft_rules -dft_configuration_mode` *`new_configuration_mode`* beforehand. Also, functional tracing can usually be done by using mode 0.

Options and Arguments

`-continue`	Specifies to continue the trace back until a primary input, complex gate, or sequential gate is reached. Additionally, the trace will terminate if a logic constant is returned for the trace back pin.
`-list`	Prints the status and pin at every trace back.
`-mode` *`integer`*	Specifies the mode for tracing back. ■ 0 does not perform constant propagation. This is usually identical or close to a functional mode trace back. ■ 1 performs tied-constant propagation ■ 2 performs tied-constant and test-mode propagation ■ 3 performs tied-constant, test-mode and shift-enable propagation. This is usually identical to the default scan shift mode. *Default*: `3`
{*`pin`* | *`subport`* | *`port`*}	Specifies the pin or port from which to start the trace back.
`-polarity`	Specifies whether to report if the polarity changed through the trace. A returned value of 0 indicates no inversion.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	A returned value of 1 indicates an inversion.
`-print`	Prints the pin and polarity at every trace back.

Examples

- Following command traces back without performing constant propagation.

```
report_dft_trace_back -mode 0 /designs/top/instances_hier/g121/pins_in/CME
/designs/top/ports_in/cme
```

- Following command traces back using the default mode. In this case a constant logic 1 value is reported.

```
legacy_genus:/> dft_trace_back /designs/top/instances_hier/g121/pins_in/CME
1
```

- Following command traces back using the default mode and requests to report whether there was a change in polarity. In this case, a constant logic 1 with no change in logic polarity is reported.

```
legacy_genus:/> dft_trace_back /designs/top/instances_hier/g121/pins_in/CME
-polarity
1 0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_flow

`report_flow`

Generates a report with the following information:

- The flow type, section list, and generation date of the templates
- The flow step order
- One line for every item in `flow_history`, indicating the success of that item and the configuration of optional flow features, and any error message

There is some overlap between `report_flow` and the summary printed at the end of `run_flow` commands. The difference is that `report_flow` reports information about the entire flow history (since the last `run_flow –reset`, if any), whereas the summary at the end of `run_flow` only includes data about the steps run by that command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_logic_levels_histogram

```
report_logic_levels_histogram [-bars integer]
     [-threshold integer] [-slack_limit float]
     [-skip_buffer] [-skip_inverter] [-cost_group]
     [-view analysis_view] [-std_cells] [-per_cost_group] [> file]
```

Prints the number of timing endpoints in the design in a histogram format.

The timing paths are grouped according to the logical depth number of the paths.

Options and Arguments

`-bars` *`integer`*	Specifies the logic level distribution histogram number. *Default*: `5`
`-cost_group`	Specifies the cost group to generate the report for.
`-details`	Reports all endpoints and logic levels.
`file`	Redirects the report to the specified file.
`-per_cost_group`	Specifies to report the logic level histogram for each cost group.
`-skip_buffer`	Excludes buffers from the logic level count.
`-skip_inverter`	Excludes inverters from the logic level count.
`-slack_limit` *`float`*	Limits the report to paths with less than the specified slack.
`-std_cells`	Specifies to create logic level depth histogram across standard cells.
`-threshold` *`integer`*	Specifies the threshold for the number of logic levels to report on. Only logic levels equal to or greater than the specified number are reported.
`-view` *`analysis_view`*	Specifies to report the histogram for the specified analysis view.

Related Information

Related command: report timing on page 709

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_metric

```
report_metric -file file
     [-format {html|json} ]
     [-id string] [-names string]]
```

Reports the statistics information (QoR metrics) that was recorded at various stages or that was read in, or compares the metrics of two specified runs.

Note: Power metrics are only recorded if you enabled the `statistics_enable_power_report` root attribute.

Options and Arguments

`-file` *`file`*	Specifies the file name to which to write the report.
`-format {html|json}`	Specifies the format type of the report.
`-id` *`string`*	Specifies the ID of the metrics for which you want to see the report.
`-names` *`string`*	Specifies the stage(s) of the run that you want to list in the report.
`string`	Specifies the pattern to match.

Related Information

Affected by these commands:

create_snapshot on page 185
read_metric on page 201
delete_metric on page 189

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_obj

```
report_obj
     [string1] [string2]
     [-all | [-settable] [-computed]
     [-categories list_of_categories]
     [-tcl | -verbose]
```

Summarizes all attributes associated with the object being reported. The default output shows the non-default root attributes which were modified either by user or in the run script, Extra customization can be added using filters to report specific attributes of interest.

The default output reports those attributes that are set to non-default values and are not computed. This makes it easy to see any changes that were made in the run scripts.

This command supports wild cards. This command is very useful in scenarios where user can understand how their flow changes with root attributes.

You can also add more attributes or filter out some attributes using various options below.

Options and Arguments

`-all`
: Reports all non-computed attributes and their current value, even if the current value matches the default value.

`-categories` *`list_of_categories`*
: Reports only those attributes that are in the given categories. Wildcards * and ? are allowed. Attributes that do not have a category are assigned the reserved category name NULL. The report puts all the attributes in a category together.

`-computed`
: Includes attributes that are computed (for example `is_computed` == `true`).

  By default, the computed attributes are not included. If the data is not computed, this parameter triggers building the timing graph, and so on, that may take significant run time. This parameter cannot be used with -`all`.

`-settable`
: Restricts the report to all attributes marked as `is_settable` that are not set to the default value. This parameter cannot be used with `-all`.

`string1`
: Reports attributes for the specified objects.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Default: `root`
`string2`	Returns only those attributes whose names match the specified pattern. You can use the * and ? wild characters.
	Default: `*`
`-tcl`	Returns the objects and their values in a Tcl-style list, that is, a list of keyword-value pairs.
`-verbose`	Returns the objects data type and default value.
	You cannot use this option with the `-tcl` option.

Example

```
legacy_genus:/designs/alu> report_obj
*****************************************************************************************
* Object: /
*****************************************************************************************

Attribute Name                                                    Current Value
-----------------------------------------------------------------------------------------
base_name                                                         /
common_ui                                                         false
elapsed_runtime                                                   75
escaped_name                                                      {}
information_level                                                9
load_average                                                      0.01
memory_usage                                                      409.48
name                                                              {}
obj_type                                                          root
peak_memory                                                       427.39
peak_physical_memory_usage                                        153.0
physical_memory_usage                                             153.0

platform_wordsize                                                 64
program_name                                                      Genus(TM) Synthesis Solution
program_version                                                   16.20-p003_1
startup_license                                                   Genus_Synthesis
tinfo_tstamp_file                                                 .rs_ria.tstamp
library                                                           tutorial.lib
current_design                                                    /designs/alu
db_units                                                          no_value
phys_assume_met_fill                                              0.0


legacy_genus:/designs/alu>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_scan_compression_logic

```
report_scan_compression_logic
     [-compactor hinst] [-testic module]
     [-highlight_in_gui] [-colors string]
     [-report_unplaced] [-report_tracing {none|summary|detail|debug}]
```

Reports the area and wirelength overhead of the scan compression logic.

Note: The wirelength overhead is only reported if the design is placed.

Options and Arguments

`-colors` *`string`*	Specifies a list of category and color pairs. Category can be any of: `SI`, `HEAD`, `TAIL`, `MASK`, `FULLSCAN`, `CODEC`, or `SO`. *Default*: `"HEAD lightblue TAIL lightgreen MASK blue FULLSCAN yellow CODEC red"`
`-compactor` *`hinst`*	Specifies the name of the scan compression logic macro. *Default*: `COMPACTOR`
`-hightlight_in_gui`	Highlights the scan compression logic wires in the GUI.
`-report_tracing {none|summary|detail|debug}`	Reports details on compression logic tracing. *Default*: `none`
`-report_unplaced`	Reports the unplaced pins and ports.
`-testic` *`module`*	Specifies the name of the test interconnect macro if present. *Default*: `TestIC`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

```
legacy_genus:/> report_scan_compression_logic -compactor COMPACTOR
Reporting statistics for scan compression macro: hinst:DLX_CORE/COMPACTOR ...
                 Area (um^2)     #Insts      #Bufs      #Nets Wirelength (um)
SI to CoDec          125.856         47          0          8         974.936
Channel Heads        636.120        128          0         64         169.622
Channel Tails         87.552         64          0         64        1706.644
CoDec                757.872        294          2        471       10314.399
Mask                1140.570        308          0        247        5603.287
Fullscan Muxing        0.684          1          1          2        1258.399
CoDec to SO           13.680          8          0          8          85.400
All                 2762.334        850          3        864       20112.687
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_signoff_timing

```
report_signoff_timing
     [-sdc sdc_files] [-netlist path] [-libraries lib_list]
     [-include file | -report_timing_command command]
     [> file]
```

Generates an Encounter Timing System timing report.

To run this command you need to have access to the Encounter ® Timing System software.

Options and Arguments

file	Specifies the name of the file to which the report must be written. *Default*: `vtim_ets_timing_rpt`
`-include` *file*	Specifies the file containing the Encounter Timing System commands to be executed. These commands will be read in after the libraries, netlist and SDC constraints have been read into the Encounter Timing System tool.
`-libraries` *lib_list*	Specifies the list of libraries to be read by Encounter Timing System. If no libraries are specified, the same libraries that were specified for synthesis are used.
`-netlist` *path*	Specifies the path to the netlist. If no netlist is specified, the netlist is generated with the `write_hdl` command in the `.vtim_ets` directory.
`-report_timing_command` *command*	Specifies a string that contains a single Encounter Timing System report command.
`-sdc` *sdc_files*	Specifies the SDC file(s) for the design. If no SDC file(s) are specified, the SDC files are generated using the `write_sdc` command in the `.vtim_ets` directory.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics

```
statistics
     {add_metric | log | read | remove_metric | report
     | reset | run_stage_ids | write}
```

Tracks the statistics information (QoR metrics) for the design.

Note: Power metrics are only recorded if you enabled the `statistics_enable_power_report` root attribute.

Options and Arguments

`add_metric`	Adds a user-defined metric to be tracked.
`log`	Computes the metrics at the stage where it is called and stores the data with the specified tag name.
`read`	Reads the metrics from the specified file into Genus.
`remove_metric`	Removes a user-defined metric.
`report`	Reports the metrics that were recorded at various stages or that were read in.
`reset`	Resets the metrics recorded or read in during the session.
`run_stage_ids`	Reports a summary of run IDs and the associated stage IDs and run description in the RC session.
`write`	Writes out the QoR metrics recorded at various stages to the specified database file

Related Information

Related commands:

statistics add_metric on page 744

statistics log on page 746

statistics read on page 748

statistics remove_metric on page 749

statistics report on page 750

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics reset on page 753

statistics run_stage_ids on page 754

statistics write on page 755

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics add_metric

```
statistics add_metric
     -name metric -function function [argument]...
     [-header | -footer}
```

Adds a user-defined metric to be tracked.

Options and Arguments

`-footer`	Specifies to add the metric at the footer of the report.
	This allows to track static and dynamic values at the end of the report.
`-header`	Specifies to add the metric at the header of the report.
	This allows to track static values at the beginning of the report.
`-name` *`metric`*	Specifies the name of the user-defined metric.
`-function` *`function`*	
	Specifies the Tcl function to execute to compute the metric.
`argument`	Specifies an argument of the Tcl function

Example

- The following example defines a Tcl function `get_state` which returns the current design state. The user-defined metric is called `state`.

```
proc get_state {} {
    return [get_attr state /designs/cscan]
}
statistics add_metric -name state -function get_state
```

- The following example defines a metric to be added to the header of the report.

```
proc get_date {} {
    set temp "[clock format [clock seconds] -format "%b%d-%T"]"
  return $temp
}
statistics add_metric -header -name Date -function get_date
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	statistics report on page 750
Related command:	statistics remove_metric on page 749

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics log

```
statistics log
     -stage_id string [-ignore_user_defined]
```

Computes the metrics at the stage where it is called and stores the data with the specified stage identifier (ID).

Options and Arguments

`-ignore_user_defined`

Specifies to ignore the user-defined metrics at this stage.

`-stage_id` *`string`*

Specifies the name of the user-defined stage.

To list the stage names already used during this run use the `statistics run_stage_ids` command.

Predefined stages are:

`elaborate`

`generic`

`global_map`

`incremental`

`place`

`incremental_place`

`spatial`

`spatial_incremental`

`synthesize`

Example

The following command computes the metrics after reading in the constraints and calls the stage `constraints`.

```
read_sdc my_constraints.sdc
statistics log -stage_id constraints
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	statistics report on page 750
	statistics run_stage_ids on page 754
Related command:	statistics run_stage_ids on page 754

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics read

```
statistics read
     -file file -reset
```

Reads the statistics information (QoR metrics) from the specified file into Genus.

You can read in multiple files, but you can only read in one file per command.

After reading in the information, the command returns similar info as the `statistics run_stage_ids` command.

Options and Arguments

`-file file`	Specifies the file to read the statistics information from.
`-reset`	Resets the existing content before reading the database file.

Example

```
legacy_genus:/> statistics read -file sample2.db
Reading file sample2.db
Sourcing './sample2.db' (Thu Aug 05 15:15:27 -0700 2010)...
Done reading file sample2.db

       Run & Stage ID summary
       ----------------------
Run ID                Stage ID(s)               Run Description
-----------------------------------------------------------------
sample2 elaborate generic global_map incremental n/a
```

Related Information

Affects this command:	statistics run_stage_ids on page 754
Related command:	statistics write on page 755
Related attributes:	statistics_run_id
	statistics_run_description

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics remove_metric

```
statistics remove_metric
     -name metric
```

Removes a previously user-defined metric.

Options and Arguments

`-name` *`metric`*	Specifies the name of the user-defined metric to be removed.

Example

The following example removes the user-defined metric `state`.

```
statistics remove_metric -name state
```

Related Information

Affects this command:	statistics report on page 750
Related command:	statistics add_metric on page 744

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics report

```
statistics report
     -run_id run_id [-compare run_id ]
     [-stage_id stage_tag [-compare_stage_id stage_tag]]
     [-ignore_user_defined] > file
```

Reports the statistics information (QoR metrics) that was recorded at various stages or that was read in, or compares the metrics of two specified runs.

Note: Power metrics are only recorded if you enabled the `statistics_enable_power_report` root attribute.

Options and Arguments

`-compare` *`run_id`*
: Specifies the run that you want to compare to the run specified with the `-run_id` option.

`-compare_stage_id` *`stage_tag`*
: Specifies the stage(s) of the run specified with the `-compare` option that you want to list in the report.

  Note: You cannot specify this option without the `-stage_id` option.

`-ignore_user_defined`
: Specifies to ignore the user-defined metrics in the report.

`-run_id` *`run_id`*
: Specifies the run for which you want to see the statistics information.

`-stage_id` *`string`*
: Specifies the stage(s) of the run specified with the `-run_id` option that you want to list in the report.

  If you omit this option, the report will show statistics information for all stages.

Examples

- The following command shows the report for run sample1 for the incremental stage.

```
statistics report -run_id sample1 -stage_id incremental
    QOR statistics summary
    ----------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
        Metric         incremental
-----------------------------------
WNS.I2O                no_value
WNS.I2C                1387.9
WNS.CLK1               2219.9
WNS.C2C                no_value
WNS.C2O                2187.2
WNS.default            no_value
TNS                    0
Violating_paths        0
runtime                0.14
memory                 0.00
Leakage_power          327.19
Net_power              362656.25
Internal_power         19369.41
Clock_gating_instances 0
total_net_length       n/a
average_net_length     n/a
routing_congestion     n/a
utilization            0.0
Inverter_count         10
Buffer_count           0
timing_model_count     0
sequential_count       16
unresolved_count       0
logic_count            13
Total_area             538.01
Cell_area              538.01
Net_area               0.00
```

- The following report includes header and footer information defined with the `statistics add_metric -header` and `-footer` options.

```
QOR statistics summary
----------------------
Machine_Info rcae010 Intel(R) Xeon(TM) CPU 2.80GHz 2793.238Mhz 32bits 2_cores
Date Aug17-21:56:48
...

        Metric         elaborate  generic  global_map incremental
-----------------------------------------------------------------
WNS.I2O                n/a       no_value  no_value   no_value
WNS.I2C                n/a       1533.5    1912.6     1387.9
WNS.CLK1               n/a       no_value  2219.9     2219.9
WNS.C2C                n/a       no_value  no_value   no_value
WNS.C2O                n/a       2187.2    2314.1     2187.2
WNS.default            no_value  no_value  no_value   no_value
TNS                    0         0         0          0
Violating_paths        0         0         0          0
...
Elapsed_Time 266
Super_Thread_Total_Runtime 245.000
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command does a comparison between two runs.

```
statistics report -run_id medium_effort -compare high_effort -stage global_map
    QOR statistics summary
    ----------------------


        Metric         global_map.medium_effort global_map.high_effort %diff
-------------------------------------------------------------------------------
WNS.I2O                no_value                 no_value                n/a
WNS.I2C                1912.6                   1887.6                  1.31
WNS.CLK1               2219.9                   2219.9                  0.0
WNS.C2C                no_value                 no_value                n/a
WNS.C2O                2314.1                   2314.1                  0.0
WNS.default            no_value                 no_value                n/a
TNS                    0                        0                       n/a
Violating_paths        0                        0                       n/a
runtime                1.74                     0.32                    81.61
memory                 8.05                     1.11                    86.21
Leakage_power          1730.88                  1730.88                 0.0
Net_power              453828.12                453828.12               0.0
Internal_power         59389.35                 59389.35                0.0
Clock_gating_instances 0                        0                       n/a
total_net_length       n/a                      n/a                     n/a
average_net_length     n/a                      n/a                     n/a
routing_congestion     n/a                      n/a                     n/a
utilization            0.0                      0.0                     n/a
Inverter_count         10                       10                      0.0
Buffer_count           0                        0                       n/a
timing_model_count     0                        0                       n/a
sequential_count       16                       16                      0.0
unresolved_count       0                        0                       n/a
logic_count            13                       13                      0.0
Total_area             995.70                   995.70                  0.0
Cell_area              995.70                   995.70                  0.0
Net_area               0.00                     0.00                    n/a
state                  mapped                   mapped                  n/a
```

Related Information

Affected by these commands:	statistics log on page 746
	statistics read on page 748
	statistics add_metric on page 744
Affected by this attribute:	statistics_enable_power_report

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics reset

```
statistics reset
```

Removes the statistics information (QoR metrics) recorded or read in during the session.

Example

```
legacy_genus:/> statistics run_stage_ids

       Run & Stage ID summary
       ----------------------
Run ID                Stage ID(s)               Run Description
-----------------------------------------------------------------
sample2 elaborate generic global_map incremental n/a
legacy_genus:/> statistics reset
Info    : Reset the statistics information preset in the database. [STAT-4]
legacy_genus:/> statistics run_stage_ids
Info    : No run and stage id data available to report. [STAT-12]
        : Either a statistics db file was not read in or the 'statistics_log_data'
attribute was not enabled for the synthesis run.
```

Related Information

Affects these commands: statistics report on page 750
statistics run_stage_ids on page 754

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics run_stage_ids

```
statistics run_stage_ids
```

Reports a summary of run IDs and the associated stage IDs and run description in the RC session.

The information is derived from the current run and from other database files that you read in.

Related Information

Affected by these commands:	statistics log on page 746
	statistics read on page 748
	statistics reset on page 753
Related attributes:	statistics_run_id
	statistics_run_description

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

statistics write

```
statistics write
     [-to_file file]
```

Writes out the statistics information (QoR metrics) recorded at various stages to the specified database file.

Note: When you specify an existing file, the tool will overwrite the existing data.

Options and Arguments

`-to_file` *`file`*	Specifies the file to which to write the statistics information. If you omit this option, the data is recorded in the file determined by the `statistics_db_file` attribute.

Example

```
legacy_genus:/> set_attribute statistics_db_file stats/test.db /
 Setting attribute of root '/': 'statistics_db_file' = stats/test.db
legacy_genus:/> statistics write
Info    : Writing statistics database to file. [STAT-3]
        : Writing to db file 'stats/test.db'
```

Related Information

Related command:	statistics read on page 748
Related attribute:	statistics_db_file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

time_info

```
time_info [-report | -current]
     [-table tableID] [-stamp]
     [-save tstampfile] [-load tstampfile]
     [stage][> file]
```

Reports and tracks runtime and memory performance.

Note: This command provides a shortcut syntax for `-report` and `-stamp` command options. For example, `time_info STAGE_6` is the shortcut for `time_info -stamp STAGE_6 -report`.

Options and Arguments

`-current`	Specifies to report upto the current stage.
`file`	Specifies the name of the file to which to write the report.
`-load` *`tstampfile`*	Specifies the name of the ".tstamp" file from where to load runtime metrics.
`-report`	Specifies to report all stages upto this point.
`-save` *`tstampfile`*	Specifies the name of the ".tstamp" file to save the runtime metrics.
`stage`	Specifies the name of the stage.
`-stamp`	Specifies to create a timestamp.
`-table tableID`	Specifies the metrics table ID to support multiple tables. *Default*: '`default`'

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

timestat

```
timestat
     [string] [> file]
```

Reports the runtime and memory used up to this stage (time that the information was requested).

Options and Arguments

string	Specifies a user-defined string which allows you to identify at which stage in the design the information was requested. *Default:* `undefined`.
file	Redirects the command output to the specified file.

Examples

- The following script extract requests the information after RTL optimization.

```
synthesize -to_generic -eff $SYN_EFF
puts "Runtime & Memory after 'synthesize -to_generic'"
timestat GENERIC
```

- The following example shows the output after mapping.

```
legacy_genus:/> timestat map
============================================
The RUNTIME after map is 0.45 secs
and the MEMORY_USAGE after map is 24.16 MB
============================================
```

- The following example shows the output if no user-defined string was specified.

```
legacy_genus:/> timestat
============================================
The RUNTIME after undefined is 0.45 secs
and the MEMORY_USAGE after undefined is 24.16 MB
============================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

validate_timing

```
validate_timing
     [-sdc sdc_files] [-netlist path] [-libs lib_list]
     [-include file | -rep_tim_str command]
     [-keep_temp_dir] [> file]
```

Generates an Encounter Timing System timing report.

To run this command you need to have access to the Encounter ® Timing System software.

Options and Arguments

file	Specifies the name of the file to which the report must be written. *Default*: `vtim_ets_timing_rpt`
`-include` *file*	Specifies the file containing the Encounter Timing System commands to be executed. These commands will be read in after the libraries, netlist and SDC constraints have been read into the Encounter Timing System tool.
`-keep_temp_dir`	Does not remove the temporary (`.vtim_ets)` directory in which the tool generates the SDC file or netlist.
`-libs` *lib_list*	Specifies the list of libraries to be read by Encounter Timing System. If no libraries are specified, the same libraries that were specified for synthesis are used.
`-netlist path`	Specifies the path to the netlist. If no netlist is specified, the netlist is generated with the `write_hdl` command in the `.vtim_ets` directory.
`-rep_tim_str` *command*	Specifies a string that contains a single Encounter Timing System report command.
`-sdc` *file_list*	Specifies the SDC file(s) for the design. If no SDC file(s) are specified, the SDC files are generated using the `write_sdc` command in the `.vtim_ets` directory.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command reads the Encounter Timing System commands to be executed from the `ets_include` file.

  ```
  validate_timing -netlist test.v -libs $env(REGLIBS)/tutorial.lib \
  -include ets_include
  ```

  Because no SDC file was specified, the `vtim_run_ets.sdc` file is generated in the `.vtim_ets` directory.

- The following command directly specifies which Encounter Timing System command to be execute.

  ```
  validate_timing -rep_tim_str "report_timing"
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

12

Physical

- check_floorplan on page 764
- check_placement on page 768
- convert_polygon_to_boxes on page 770
- create_clip on page 771
- create_group on page 772
- create_place_blockage on page 773
- create_placement_blockage on page 775
- create_placement_halo_blockage on page 777
- create_region on page 778
- create_route_blockage on page 780
- create_routing_blockage on page 782
- create_routing_halo_blockage on page 784
- create_row on page 785
- create_single_row on page 787
- create_track on page 789
- cut_power_domain_by_overlaps on page 790
- def_move on page 791
- delete_rows on page 792
- duplicate_register on page 793
- generate_ple_model on page 795
- generate_reports on page 797

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- get_computed_shapes on page 799
- get_obj_in_area on page 800
- modify_power_domain_attr on page 801
- move_blockage on page 803
- move_instance on page 805
- move_port on page 806
- move_region on page 808
- place_dft on page 809
- place_dft_sequentials on page 810
- predict_floorplan on page 811
- read_clip on page 812
- read_def on page 814
- read_encounter on page 818
- read_physical on page 819
- read_sdp_file on page 821
- read_spef on page 822
- report congestion on page 823
- report utilization on page 824
- resize_blockage on page 825
- resize_region on page 826
- restore_congestion_map on page 827
- specify_cell_pad on page 828
- specify_floorplan on page 829
- summary_table on page 831
- update_congestion_map on page 833
- update_group on page 834
- update_instance_data on page 835

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- write_congestion_map on page 836
- write_def on page 837
- write_lib_lef on page 839
- write_sdp_file on page 840
- write_spef on page 841

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_floorplan

```
check_floorplan
     [-design design]
     [-macros | -ports | -utilization] [-suppress_utilization]
     [-detailed -max_per_violation integer]
     [-status [-skip_ports]] [-report_utilization]
```

Checks for the validity of the floorplan.

You can use this command in a script to ensure that there is some valid physical information available before you continue with the next steps in the flow. Run this command after reading the floorplan and before performing physical synthesis.

Options and Arguments

`-design` *`design`*
: Specifies the name of the design for which to do the checking.

`-detailed`
: Reports a detailed violation report.

`{-macros | -ports | -utilization}`
: Limits the checking of the design to one of the following:
- macro placement—unplaced macros will be flagged.
- port placement—-unplaced ports will be flagged.
- unusual utilization

`-max_per_violation` *`integer`*
: Specifies the number of objects to print per violation.

Default: `10`

`-report_utilization`
: Reports details about the areas with unusual utilization.

`-skip_ports`
: Skips the port information when you request the status information.

`-status`
: Specifies to return the status (0 or 1) of the command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The command returns 0 in the following circumstances:

- no DEF file was read
- unplaced macros are present
- unplaced ports are present

If this option is omitted, the command generates a report listing the unplaced macros and ports.

`-suppress_utilization`

Suppresses utilization checks.

Useful when running `check_floorplan` on an elaborated design.

Examples

The following example shows the `status` of the floorplan

```
legacy_genus:/>  check_floorplan -status
0
```

- The following command checks for unplaced macros.

```
legacy_genus:/> check_floorplan -macros


==============================================================================
  Generated by: Genus Synthesis Solution version
  Generated on: date
  Module:       rct
==============================================================================

  =======================
  check_floorplan SUMMARY
  =======================

-------------------------------------------------------------------------------------------------------
  Floorplan Object | Status | #Violations |   #Total    |   ID   |  Violation Reason
-------------------------------------------------------------------------------------------------------
  Core/Die Box     | PASS   |          0  |          1  |        |
  Cell Sites       | PASS   |          0  |         40  |        |
  Hard macros      | WARN   |          1  |          3  | FPLN-5 | not pre-placed and fixed
-------------------------------------------------------------------------------------------------------

The checker finds : 1 category does not pass the checking.
Please check it and make necessary corrections before proceeding with your synthesis job.

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s
```

- The following command reports the areas with unusual utilization in the design.

```
legacy_genus:/> check_floorplan -utilization


==============================================================================
  ...
  Module:       rct
==============================================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
  =======================
  check_floorplan SUMMARY
  =======================

-------------------------------------------------------------------------------------------------
 Floorplan Object| Status | #Violations| #Total |   ID  |  Violation Reason
-------------------------------------------------------------------------------------------------
 Core/Die Box    | PASS   |          0 |      1 |       |
 Cell Sites      | PASS   |          0 |     40 |       |
 Top/Regions Util| WARN   |          4 |      4 | FPLN-6| cells out of bound, over- or under-utilized
-------------------------------------------------------------------------------------------------

The checker finds : 1 category does not pass the checking.
Please check it and make necessary corrections before proceeding with your synthesis job.

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s
```

- The following command reports the details of the areas with unusual utilization.

```
legacy_genus:/> check_floorplan -utilization -report_utilization

==============================================================================
  ...
  Module:       rct
==============================================================================

  ====================================
  check_floorplan UTILIZATION REPORT
  ====================================

  Fence REGION_FOUR
  -------------------------------------------------------------------------
  Bounding box                      : (100.000 4.000 110.000 18.000)
  Total placeable area              :          104.720
  Fixed and placed cell area        :            0.000 (utilization  0.0%)
  Number of fixed and placed cells :                 0
  Placeable cell area               :            0.000 (utilization  0.0%)
  Number of placeable cells         :                 0
  Floorplan utilization             :              0.0%
  -------------------------------------------------------------------------

  Fence REGION_TWO
  -------------------------------------------------------------------------
  Bounding box                      : (30.000 30.000 50.000 50.000)
  Total placeable area              :          246.400
  Fixed and placed cell area        :            0.000 (utilization  0.0%)
  Number of fixed and placed cells :                 0
  Placeable cell area               :            0.000 (utilization  0.0%)
  Number of placeable cells         :                 0
  Floorplan utilization             :              0.0%
  -------------------------------------------------------------------------

  ....


  =======================
  check_floorplan SUMMARY
  =======================
 .....

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command shows the detailed violation list.

```
legacy_genus:/> check_floorplan -detailed
================================================================================
  ...
  Module:       rct
================================================================================

  ====================================
  check_floorplan VIOLATION DETAILS
  ==================================
Info    : Not all ports are pre-placed and fixed. [FPLN-3]
        : The quality of physical-aware mapping might be degraded due to unpredictable placement resu
lt.
        : The following primary ports are not pre-placed and fixed.
            dataout[0]
            dataout[1]
            dataout[2]
            dataout[3]
            dataout[4]
            dataout[5]
            dataout[6]
            dataout[7]
            dataout[8]
            dataout[9]
            .....(61 more objects not listed)
Info    : Not all hard macros are pre-placed and fixed. [FPLN-5]
        : The quality of physical-aware mapping might be degraded due to
unpredictable placement result.
        : The following hard macro cell is not pre-placed and fixed.
            ram_sort
Info    : Data out of ordinary for floor plan constraints. [FPLN-6]
        : Examine it and make necessary change in order to achieve good placement
result.
        : The following floorplan constraints have extreme utilizations or invalid
data.
            Low utilization (  0.0% ) for Fence REGION_FOUR.
            Low utilization (  0.0% ) for Fence REGION_TWO.
            Low utilization (  0.0% ) for Region REGION_ONE.
            High utilization ( 93.2% ) for Top Design.


  =======================
  check_floorplan SUMMARY
  =======================

-------------------------------------------------------------------------------------------------
 Floorplan Object | Status| #Violations| #Total |   ID  |  Violation Reason
-------------------------------------------------------------------------------------------------
 Core/Die Box     | PASS  |          0 |      1 |       |
 Cell Sites       | PASS  |          0 |     40 |       |
 Primary Ports    | WARN  |         71 |     71 | FPLN-3| not pre-placed and fixed
 Hard macros      | WARN  |          1 |      3 | FPLN-5| not pre-placed and fixed
 Top/Regions Util | WARN  |          4 |      4 | FPLN-6|cells out of bound, over- or under-utilized
 Blockages        | PASS  |          0 |     11 |       |
-------------------------------------------------------------------------------------------------

The checker finds : 3 categories do not pass the checking.
Please check them and make necessary corrections before proceeding with your synthesis job.

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_placement

```
check_placement design
     [-file file] [-verbose]
```

Checks the placement legality and highlights illegal objects. The command also returns some instance status statistics.

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`-file file`	Specifies the file to which to write the report.
`-verbose`	Specifies to generate a verbose report.
design	Specifies the name of the design for which to create the group.

Examples

- The following example shows the non-verbose output of the `check_placement` command send to the console:

```
legacy_genus:/designs/DTMF_CHIP/> check_placement [find / -design DTMF*]
   Instance status statistics :
       cover   : 0
       fixed   : 71
       placed  : 0
       unplaced: 5906
Number of placement errors : 0

Legality check : OK
```

- The following example shows an extract of the verbose output of the `check_placement` command:

```
legacy_genus:/designs/DTMF_CHIP/physical/rows> check_placement [find / -design
DTMF*] -verbose
Info    : Placement Information. [PLC-1]
        : [I] Report Inst status stats...
Info    : Placement Information. [PLC-1]
        : [I] def / nl...0xd0d6b10 / 0xd0b3040
   Instance status statistics :
       cover   : 0
       fixed   : 71
       placed  : 0
       unplaced: 5906
Info    : Placement Information. [PLC-1]
        : [I] Checking legality...
Info    : Placement Information. [PLC-1]
        : [I] check (127)
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Info    : Placement Information. [PLC-1]
        : [I] checking overlaps...
Info    : Placement Information. [PLC-1]
        : [I] Obstruction rect (771280,297570 - 1097045,442015)
Info    : Placement Information. [PLC-1]
        : [I] Found 0 polygons and 1 rects.
Info    : Placement Information. [PLC-1]
        : [I] Obstruction rect (757845,512370 - 892440,1071790)
Info    : Placement Information. [PLC-1]
        : [I] Found 0 polygons and 1 rects.
....
Info    : Placement Information. [PLC-1]
        : [I] writing summary...
Number of placement errors : 0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

convert_polygon_to_boxes

`convert_polygon_to_boxes` *`pt pt pt`* `[`*`pt`*`]...`

Converts a polygon into a list of boxes.

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`pt pt pt [pt]...`	Specifies a list of the coordinates of at least three points that define the region polygon. Use floats for the coordinates.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_clip

```
create_clip
     -outdir string -instances {inst|hinst}...
     [-logical_only] [design]
```

Generates timing constraints and physical data for a design clip, that allow each unit (clip) to be synthesized separately but with the full design context.

These data should be generated after an initial synthesis of the full design.

Options and Arguments

design	Specifies the name of the design. This parameter is only required if multiple designs are loaded.
-instances {*inst*|*hinst*}	Specifies a list of instances that identify the clip.
-logical_only	Skips the generation of the physical data.
-outdir *string*	Specifies the directory in which to generate the data.

Example

Following is the suggested flow usage.

```
source setup.tcl
read_hdl all_hdl_files
elaborate top_design
read_sdc top_design.sdc
read_def def_file
set_db <block_units>.boundary_opto false
syn_generic -physical
syn_map -physical
syn_opt -physical
generate_units ...
```

Related Information

Affects this command: read_clip on page 812

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_group

```
create_group group
     instance_list
     [-region region] [-design design] [-overwrite]
```

Creates a group of instances that must be placed close together.

The command returns the path to the `group` object that it creates. You can find the objects created by the `create_group` command in:

`/designs/design/physical/groups/`

The user-defined name is stored in the `def_name` group attribute.

Options and Arguments

`-design design`	Specifies the name of the design for which to create the group.
`-overwrite`	Overwrites any previously existing group membership.
`-region name`	Associates the specified region with the group.
group	Specifies the name of the group to be created.
instance_list	Specifies the instances that belong to this group.

Example

The following example creates group `gr_reg` with two instances:

```
create_group mygroup -region myreg  [find / -inst a_reg_3] [find / -inst a_reg_5]
```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related command:	create_region on page 778

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_place_blockage

```
create_place_blockage [-name name]
     [-boxes {llx lly urx ury}...]
     [-component name]
     [-pushdown] [-soft]
     [-partial [-exclude_flops]]
     [-density float]
     [-design design]
     [-no_update]
```

Creates a placement blockages used to block standard cell placement. You can use the `edit_netlist delete` command to delete placement blockages.

Options and Arguments

`-boxes {llx lly urx ury}...`	Specifies the lower left and upper right coordinates of the blockage box. Use floats for the coordinates.
`-component name`	Associates the blockage with the specified component.
`-density float`	Specifies the percentage of the blockage area to be used for standard cells during initial placement. Enter a value from `0.0` to `0.95`.
`-exclude_flops`	Prevents flops to be placed in a partial placement blockage.
`-name name`	Specifies the name of the blockage.
`-component name`	Associates the blockage with the specified component.
`-no_update`	Prevents updating of the internal data structures. When you need to create a large number of blockages, you can add the last blockage without the `-no_update` option to update the internal data structures in one operation. This will speed up the operation.
`-partial`	Creates a partial placement blockage.
`-pushdown`	Creates a pushdown placement blockage.
`-soft`	Creates a soft placement blockage.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_placement_halo_blockage on page 777
	move_blockage on page 803
	resize_blockage on page 825
	edit_netlist delete on page 1328

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_placement_blockage

```
create_placement_blockage [-name name]
     [-boxes {llx lly urx ury}...]
     [-component name]
     [-pushdown] [-soft]
     [-partial [-exclude_flops]]
     [-density float]
     [-design design]
     [-no_update]
```

Creates a placement blockage.

Creates cell placement blockages that can be placed even outside the core area. A placement blockage is a floorplan object used to block standard cell placement.

Options and Arguments

`-boxes {llx lly urx ury}...`	Specifies the lower left and upper right coordinates of the blockage box. Use floats for the coordinates.
`-component name`	Associates the blockage with the specified component.
`-density float`	Specifies the percentage of the blockage area to be used for standard cells during initial placement. Enter a value from `0.0` to `0.95`.
`-exclude_flops`	Prevents flops to be placed in a partial placement blockage.
`-name name`	Specifies the name of the blockage.
`-component name`	Associates the blockage with the specified component.
`-no_update`	Prevents updating of the internal data structures. When you need to create a large number of blockages, you can add the last blockage without the `-no_update` option to update the internal data structures in one operation. This will speed up the operation.
`-partial`	Creates a partial placement blockage.
`-pushdown`	Creates a pushdown placement blockage.
`-soft`	Creates a soft placement blockage.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_placement_halo_blockage on page 777
	move_blockage on page 803
	resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_placement_halo_blockage

```
create_placement_halo_blockage
     -left integer -right integer
     -top integer -bottom integer
     [-soft] instance
```

Creates a placement halo around a block. A placement halo is an area that prevents the placement of blocks and standard cells within the specified halo distance from the edges of a hard macro to reduce congestion. Use this command during or after floorplanning.

Options and Arguments

`-bottom` *integer*	Specifies the halo width at the bottom of the instance. Specify the distance in DB units.
`-left` *integer*	Specifies the halo width at the left of the instance. Specify the distance in DB units.
`-right` *integer*	Specifies the halo width at the right of the instance. Specify the distance in DB units.
`-soft`	Creates a soft placement blockage.
`-top` *integer*	Specifies the halo width at the top of the instance. Specify the distance in DB units.
instance	Specifies the instance around which the placement halo is created.

Note: One DB unit is 1/1000 micron.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_placement_blockage on page 775
	move_blockage on page 803
	resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_region

```
create_region region
     [ -fence | -guide ] [-no_update]
     {-boxes {llx lly urx ury}...| -polygon pt pt pt pt [pt]...}
     [-design design]
```

Creates a region with the specified name.

The command returns the path to the `region` object that it creates. You can find the objects created by the `create_region` command in:

`/designs/design/physical/regions/`

The user-defined name is stored in the `def_name` region attribute.

Options and Arguments

`-boxes {llx lly urx ury}...`	Specifies the lower left and upper right coordinates of the box(es) that define the region. Use floats for the coordinates.
`-design design`	Specifies the name of the design for which to create the region.
`-fence`	Specifies to create a region of type fence.
`-guide`	Specifies to create a region of type guide.
`-no_update`	Prevents an update of the GUI when the new region is created.
`-polygon pt pt pt pt [pt]...`	Specifies a list of the coordinates of at least four points that define the region polygon. Use floats for the coordinates.
`region`	Specifies the name of the region to be created.

Examples

- The following command creates a box-shaped region.

  ```
  create_region GUIDE1 -guide -polygon  {0 0 10 0 10 10 0 10}
  ```

- The following command creates an L-shaped region.

  ```
  create_region GUIDE2 -guide -polygon  {0 0 0 100 20 100 20 20 100 20 100 0}
  ```

- The following command creates region `my_reg` of type `fence` for design `DTMF`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
create_region myreg -fence -box {200 200 1000 1000} [find / -design DTMF]
```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	move_region on page 808
	resize_region on page 826
Related Attributes:	Region Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_route_blockage

```
create_route_blockage [-name name]
     [-layer layer] [-cut_layer cut_layer]
     -boxes {llx lly urx ury}...
     [-slots] [-fills] [-pushdown]
     [-component name] [-design design]
```

Creates a routing blockage on the specified layer.

Options and Arguments

`-boxes {llx lly urx ury}...`

Specifies the lower left and upper right coordinates of the blockage box. Use floats for the coordinates.

`-component name`

Associates the blockage with the specified component.

`-cut_layer cut_layer`

Associates the blockage with the specified cut layer.

`-design name`

Specifies the name of the design for which to create the blockage.

`-fills`

Prevents metal fills on the specified layer in the specified areas.

`-layer layer`

Associates the blockage with the specified routing layer or list of routing layers.

The layer object must be found in the `/designs/design/physical/layers` directory.

`-no_update`

Prevents updating of the internal data structures.

When you need to create a large number of blockages, you can add the last blockage without the `-no_update` option to update the internal data structures in one operation. This will speed up the operation.

`-pushdown`

Specifies that the routing blockage must be pushed down from the top level.

`-slots`

Prevents slots on the specified layer in the specified areas.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_routing_halo_blockage on page 784
	move_blockage on page 803
	resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_routing_blockage

```
create_routing_blockage [-name name]
     [-layer layer] [-cut_layer cut_layer]
     -boxes {llx lly urx ury}...
     [-slots] [-fills] [-pushdown]
     [-component name] [-design design]
```

Creates a routing blockage on the specified layer.

Options and Arguments

`-boxes {llx lly urx ury}...`
: Specifies the lower left and upper right coordinates of the blockage box. Use floats for the coordinates.

`-component name`
: Associates the blockage with the specified component.

`-cut_layer cut_layer`
: Associates the blockage with the specified cut layer.

`-design name`
: Specifies the name of the design for which to create the blockage.

`-fills`
: Prevents metal fills on the specified layer in the specified areas.

`-layer layer`
: Associates the blockage with the specified routing layer or list of routing layers.

  The layer object must be found in the `/designs/design/physical/layers` directory.

`-pushdown`
: Specifies that the routing blockage must be pushed down from the top level.

`-slots`
: Prevents slots on the specified layer in the specified areas.

Related Information

Affects these commands: syn_generic -physical

syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related commands:

create_routing_halo_blockage on page 784
move_blockage on page 803
resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_routing_halo_blockage

```
create_routing_halo_blockage
     -min_layer layer -max_layer layer
     -distance integer
     instance
```

Adds a routing halo around a block. A routing halo restricts wire routing within the specified area for specific layer(s).

Use this command during or after floorplanning.

Options and Arguments

`-distance` *`integer`*	Specifies the width of the halo around the specified instance. Specify the width in DB units
`instance`	Specifies the instance around which the routing halo is created.
`-min_layer` *`layer`*	Specifies the lowest routing layer for which to create the halo blockage.
`-max_layer` *`layer`*	Specifies the highest routing layer for which to create the halo blockage.

Note: One DB unit is 1/1000 micron.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
	syn_opt -spatial
Related command:	create_routing_blockage on page 782
	move_blockage on page 803
	resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_row

```
create_row row
     -site site_name
     [-flip_first_row | -no_flip_rows]
     [-no_abut | -no_abut_first ]
     [-limit_rows_in_core]
     [-spacing  distance ]
     [-area {x1 y1 x2 y2} | -rects {{x1 y1} {x2 y2}...}
     | -polygon {{x1 y1} {x2 y2}...}]
```

Creates rows for the specified site. You can specify the row boundary or use the core area. This command can create overlapping rows. This command can only create horizontal rows. By default, the rows are flipped and abutted.

The following restrictions apply:

- Rows can be created both inside and outside the core area, but within the die.
- All new rows created in an area must be an integer multiple of any existing rows in the same area.
- Non-integer multiple height rows cannot be created outside power domains.

You can only use this command after you read the design.

Options and Arguments

`-area {x1 y1 x2 y2}`

Specifies the coordinates of the area in which the rows must be created.

- x1 specifies the lower left x coordinate of the area.
- y1 specifies the lower left y coordinate of the area.
- x2 specifies the upper right x coordinate of the area.
- y2 specifies the upper right y coordinate of the area.

If you do not specify this option, the core area is used by default.

`-flip_first_row`

Specifies that the first row is flipped mirrored to its x-axis.

This option cannot be used with the `-no_flip_rows` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-limit_rows_in_core`

Specifies that new rows must be limited inside the core.

`-no_abut`

Prevents that rows are abutted (created back to back).

By default, the rows are abutted.

`-no_abut_first`

Specifies that the first row should not be abutted.

This option cannot be used with the `-no_abut` option.

`-no_flip_rows`

Prevents that the rows are flipped.

By default, the rows are flipped.

`-polygon {{x1 y1} {x2 y2}...}`

Specifies the coordinates of the polygon in which the rows must be created.

`-rects {{x1 y1} {x2 y2}...}`

Specifies the coordinates of the boxes in which the rows must be created.

`-spacing float`

Specifies the spacing between rows.

Default: `0.0 um`

`-site site_name`

Specifies the name of the site to be used for creating the rows.

Example

The following command creates rows for the site lt3site within the area defined by the coordinates 376.7, 577.57, 688.94, 725.68, maintaining a spacing of 1 micron between the rows, without the first row being abutted:

```
create_row -site lt3site -area 376.7 577.57 688.94 725.68 -spacing 1.0 \
-no_abut_first
```

Related Information

Affects these commands: syn_generic -physical

syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_single_row

```
create_single_row row
     -macro string
     -orientation {N|S|E|W|FN|FS|FE|FW}
     -llx integer -lly integer
     -height integer -width integer
     [-no_update] [-design design]
```

Creates a row with the specified name. Coordinates and dimensions must be specified in DB units, where one DB unit is 1/1000 micron.

The command returns the path to the `row` object that it creates. You can find the objects created by the `create_row` command in:

`/designs/design/physical/rows/`

Options and Arguments

`-design design`	Specifies the name of the design for which to create the region.
`-height integer`	Specifies the height of the row.
`-llx integer`	Specifies the lower left X coordinate of the row.
`-lly integer`	Specifies the lower left Y coordinate of the row.
`-macro name`	Specifies the name of the LEF site to be used for the row.
`-no_update`	Prevents an update of the GUI when the new row is created.
`-orientation`	Specifies the orientation of the row to be created.
`-width integer`	Specifies the width of the row.
`row`	Specifies the name of the row to be created.

Example

The following command creates row `my_row` for design `DTMF`.

```
create_single_row my_row -macro core_site -orientation N -llx 250 -lly 260 \
-width 900 -height 5 -design [find / -design DTMF]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related Attributes:	Row Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_track

```
create_track
     -start integer -step integer -num integer
     -layer string... [-vertical] [-design design]
```

Creates physical tracks (or routing grid) for the specified layer. This command creates non-uniformed, customized tracks in selected design area. The first track is located at an offset from the placement grid set by the OFFSET value for the layer in the LEF file. The track spacing is the PITCH value for the layer defined in LEF.

Options and Arguments

`-design` *design*	Specifies the name of the design in which the tracks must be created.
`-layer` *string*	Specifies the routing layer for which the tracks are created.
`-num` *integer*	Specifies the number of tracks to be created.
`-start` *integer*	Specifies the starting X or Y coordinate. X coordinate applies to vertical tracks Y coordinate applies to horizontal tracks.
`-step` *integer*	Specifies the spacing between the tracks. If not specified, step value will be the pitch of the layer defined in the techfile.
`-vertical`	Specifies to create vertical tracks. By default, a horizontal track is created.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

cut_power_domain_by_overlaps

```
cut_power_domain_by_overlaps
     power_domain
```

In case of overlapping power domains, cuts the specified outer power domain in a donut shape.

Note: Overlapping power domains are not allowed for floorplanning. So, the inner power domain must be cut out from the outer power domain.

Options and Arguments

power_domain	Specifies the name of the outer power domain.

Related Information

Related command: check_power_domain on page 1278

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

def_move

```
def_move
     [-initialize]
     [-highlight]
     [-min_distance string]
```

Highlights cell movement in the physical tab of the GUI.

Use this command after you run the `syn_opt -physical` command.

Options and Arguments

`-highlight`	Highlights the cell movement with respect to their last stored location.
`-initialize`	Stores the location of the instances at the time the command is given with this option.
`-min_distance` *`string`*	Limits the highlighting to cells that have been moved more than the specified distance. Specify the distance in microns. If this option is omitted, highlighting is limited to those cells whose x and y locations both changed by a value greater than or equal to the row height.

Example

Following illustrates the usage:

```
def_move -initialize
syn_opt -physical -incr
def_move -highlight
```

Related Information

Related command: syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_rows

```
delete_rows
     row [row]...]
```

Deletes specified rows.

Options and Arguments

string	Specifies the row(s) to be deleted.

Related Information

Related command:	create_row on page 785

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

duplicate_register

```
duplicate_register instance_list
     [ [-num_copies integer] [-fanout_list {{pin|port}...}]...
     | -timing integer]
     [-verbose] [-local] [-no_cross] [-score]
```

Duplicates the specified register(s) to reduce the load on the register outputs and thus improving the timing. The duplicated registers will be part of the same hierarchy as the original register. Duplication is controlled by the specified options. You can run this command any time after the design has been elaborated. If the design has been placed, the placement of the leaf instances in the fanout will be taken into account.

Options and Arguments

`-fanout_list {pin|port}...`

Specifies a list of fanouts (pins or ports) that require a dedicated register. You can specify the option multiple times.

Note: The endpoints do not have to be input pins of registers.

instance_list

Specifies the list of registers to be duplicated.

`-local`

Only allows duplication of the registers if the leaf instances in the fanout of the register also belong to the same hierarchy as the original unduplicated register.

`-no_cross`

Only allows duplication of the registers if there is no buffer or inverter tree between the register and the leaf instances.

`-num_copies`

Specifies the number of copies that must be made of each specified register.

`-score`

Only allows duplication of the registers if it improves the timing of the design.

`-timing integer`

Specifies the number of loads with the worst slacks for which a dedicated register must be used.

Default: `0`

`-verbose`

Prints information about the duplicated registers.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command requests a dedicated register for two groups of fanouts.

```
duplicate_register -fanout_list {U1/U1/g1/AN U1/U1/g2/AN U1/U1/g3/AN} \
-fanout_list {U2/g10/AN U2/g12/AN} U1/wr_add_reg -verbose
```

- The following command requests a dedicated register for the six loads with the worst slack.

```
duplicate_register -timing 6 U1/wr_add_reg -verbose
```

Related Information

Affected by this attribute: boundary_opto

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

generate_ple_model

```
generate_ple_model design -outfile file
```

Creates PLE correlation data for the design and stores this data in the specified file.

Net capacitances and resistances depend on technology parameters as well as the floorplan. This command refines the PLE parameters by taking both these variables into account and by comparing the PLE data with the SPEF data from Innovus. This results in a highly customized PLE equation for the given design and technology libraries.

The generated file is an encrypted file that contains

- Average Capacitance and Resistance values based on placement and default routing
- Adjustments for PLE equation parameters

The header of the generated file is readable. Check the header against the current design data to avoid mis-correlation. The header might look like:

```
# DESIGN NAME: DTMF_CHIP
# TECHNOLOGY LEF: all.lef
# CAP-TABLE: typical.captbl
# CAP SCALE: 1.0
# RES SCALE: 1.0
# ASPECT RATIO: 0.9814
```

If the design data is not inconsistent, the following message will be issued:

```
Warning : Inconsistent data. [PHYS-600]
        : Original design "DTMF_CHIP" not found in current session.
        : Input data used to create PLE correlation file is different from data
used in this session. This might lead to invalid results. Check design data.
```

Tip

Run this command once to generate PLE correlation data separately for each design. Source the generated file for every subsequent run that starts from RTL.

Note: This command does not require a floorplan, though having a good floorplan is highly recommended.

Options and Arguments

design	Specifies the design for which to generate the PLE data.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-outfile` *`file`*	Specifies the name of the output file in which to store the PLE correlation data.

Example

In the beginning of the synthesis process, use the following flow to generate the PLE data:

```
set_attribute library library_list /
set_attribute lef_library lef_files /
read_hdl hdl_file
elaborate
read_def def_file
generate_ple_model design -outfile ple_file
```

In subsequent sessions use the following flow to load the PLE data:

```
set_attribute library library_list /
set_attribute lef_library lef_files /
read_hdl hdl_file
elaborate
decrypt ple_file
...
```

Related Information

Affects these commands:	syn_generic on page 526
	syn_map on page 529
	syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

generate_reports

```
generate_reports -outdir path -tag string
     [-innovus] [-levels_of_logic]
```

Generates the QoS statistics at any stage in the flow, including timing, area, instance count, utilization and power.

Note: To create a summary table of the QoS statistics, use the `summary_table` command.

Note: To disable power reporting set the `qos_report_power` root attribute to `false`.

Options and Arguments

`-innovus`	Specifies to use the QoS statistics from the latest Innovus run. By default, the statistics are used from Genus.
`-levels_of_logic`	Prints the number of gates on the critical path per cost group in the QoR report.
`-outdir` *path*	Specifies the path to the directory where the output data should be stored.
`-tag` *string*	Specifies the prefix for reports generated at this stage.

Example

Assume the following `generate_reports` are executed.

```
legacy_genus:/> generate_reports -outdir TEST_OUT -tag initial
...
legacy_genus:/> generate_reports -outdir TEST_OUT -tag generic
...
legacy_genus:/> generate_reports -outdir TEST_OUT -tag map
```

Examining the `TEST_OUT` directory, you'll see the following reports were generated:

```
final.rpt
generic_area.rpt
generic_gates.rpt
generic_qor.rpt
generic_time.rpt
incremental_area.rpt
incremental_gates.rpt
incremental_qor.rpt
incremental_time.rpt
initial_area.rpt
initial_gates.rpt
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
initial_qor.rpt
initial_time.rpt
map_area.rpt
map_gates.rpt
map_qor.rpt
map_time.rpt
```

Related Information

Affects this command:	summary_table on page 831
Affected by this attribute:	qos_report_power

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_computed_shapes

```
get_computed_shapes
     [-output string] [-step string]
     [-maxPoint integer] [-d] [string]
```

This command is used for shape manipulation.

Options and Arguments

`-output` *`string`*	Specifies the output format. It can be one of the following: {polygon rect hrect area} *Default*: rect
`-step` *`string`*	Specifies the step size if the output format is rect or hrect.
`-maxPoint` *`integer`*	Specifies the maximum number of points for output in a polygon. Minimum is 5 and maximum is 8000
`-d`	Specifies the unit of measurement. User specified values and return values are in database units. *Default*: um
`string`	Provides a list of shapes as per the following syntax: a polygon `{{x y} {x y} ...}`, `rect {xl yl xu yu}`. You can also provide a list of polygons and/or rects `{{{x1 y1} {x2 y2} ...} {xl yl xu yu} ...}` Use the following operators on the list: AND | ANDNOT | OR | XOR | SIZE | MOVE | BBOX | INSIDE | OUTSIDE | ENCLOSE | STRADDLE | HOLES | NOHOLES

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_obj_in_area

```
get_obj_in_area
     -area float...
     [-dbu]
     -obj_type {bump|inst|pcell|pin|port|place_blockage|route_blockage|row}
     [design]
```

This command returns a list of objects inside a defined area.

Options and Arguments

`-area float`	Specifies the lower left and upper right coordinates of the search area in the format: `{llx lly urx ury}` Use floats for the coordinates. *Default*: microns
`-dbu`	Signifies that the specified values are in Database Units.
`-obj_type {bump|inst|pcell|pin|port|place_blockage|route_blockage|row}`	Specifies the object type to return.
design	Specifies the name of the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

modify_power_domain_attr

```
modify_power_domain_attr power_domain [-design design]
     [-rs_exts {T B L R}] [-min_gaps {T B L R}]
     [-box_list {llx lly urx ury}...]
     [-disjoint_hinst_box_list {{hier_inst_list region}...}]
```

Creates or modifies the physical boundary for the specified power domain.

Note: Braces in bold must be entered literally!

Options and Arguments

`-box_list {llx lly urx ury}`

Specifies a list of boxes (rectangular areas) within the design area that serve as the boundary for the power domain. For each box, specify the coordinates of the lower left and upper right corners. Use floating values.

`-design design`

Specifies the name of the design to which the power domain belongs.

This option is only required if multiple designs are loaded.

`-disjoint_hinst_box_list {{hier_inst_list region}...}`

Associates a list of hierarchical instances with a region in which they can be placed. You can specify multiple combinations.

`-min_gaps {T B L R}`

Defines the distance, in microns, that must be reserved from the power domain boundary edges for power routing. Specify a list of four floating values, one for each edge.

power_domain

Specifies the name of the power domain for which the physical boundary is created or modified. The name must correspond to the power domain name given in CPF.

`-rs_exts {T B L R}`

Specifies the boundary for legal targets to be used by the power planning and routing commands, in conjunction with the power domain boundary. Specify a list of four floating values, one for each edge.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command creates the boundary for power domain pd_08v. The domain boundary consists of 2 disjoint boxes. Hierarchical instances are associated with each box.

```
modify_power_domain_attr [find / -power_domain pd_08v] \
    -box_list [concat {0.0 107.0 21.6 116.4} {29.6 107.0 35.0 246.2}] \
    -rs_exts {1.1 2.3 3.5 4.7} -min_gaps {10.1 20.2 30 30} \
    -disjoint_hinst_box_list {{{m2/b1 m2} {0.0 107.0 21.6 116.4}} \
    {{m2 m2/b1} {{0.0 107.0 21.6 116.4} {29.6 107.0 35.0 246.2}}}}
```

Related Information

Sets these attributes: Power Domain attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

move_blockage

```
move_blockage blockage
     [-x integer | -dx integer]
     [-y integer | -dy integer]
```

Moves the specified blockage. You can specify either absolute coordinates, or a delta change in the coordinates. Coordinates or changes in coordinates must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-dx` *`integer`*	Specifies to move the blockage over the specified distance in the x-direction.
`-dy` *`integer`*	Specifies to move the blockage over the specified distance in the y-direction.
`-x` *`integer`*	Specifies the new x-coordinate for the lower left corner of the blockage.
`-y` *`integer`*	Specifies the new y-coordinate for the lower left corner of the blockage.
`blockage`	Specifies the name of the blockage to be moved.

Examples

- The following command moves blockage `my_blck` over 20 DB units in the vertical direction:

  ```
  move_blockage -dy 20 [find . -blockage my_blck]
  ```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_placement_blockage on page 775
	create_placement_halo_blockage on page 777

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

resize_blockage on page 825

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

move_instance

```
move_instance instance
     [-x integer | -dx integer]
     [-y integer | -dy integer] [-dbu]
```

Moves the specified instance. You can specify either absolute coordinates, or a delta change in the coordinates. Coordinates or changes in coordinates must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-dbu`	Specifies to use DB Units for coordinates.
`-dx` *integer*	Specifies to move the instance over the specified distance in the x-direction.
`-dy` *integer*	Specifies to move the instance over the specified distance in the y-direction.
`-x` *integer*	Specifies the new x-coordinate for the lower left corner.
`-y` *integer*	Specifies the new y-coordinate for the lower left corner.
instance	Specifies the name of the instance to be moved.

Examples

- The following command moves instance PLLCLK_INST over 2 DB units in the horizontal direction:

  ```
  move_instance -dx 2 [find . -inst PLLCLK_INST]
  ```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

move_port

```
move_port port
     [-x integer | -dx integer]
     [-y integer | -dy integer]
     [-placement_status string] [-dbu]
```

Moves the specified port. You can specify either absolute coordinates, or a delta change in the coordinates. Coordinates or changes in coordinates must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-dbu`	Specifies to use DB Units for coordinates.
`-dx` *integer*	Specifies to move the port over the specified distance in the x-direction.
`-dy` *integer*	Specifies to move the port over the specified distance in the y-direction.
`-x` *integer*	Specifies the new x-coordinate for the center of the port.
`-y` *integer*	Specifies the new y-coordinate for the center of the port.
port	Specifies the name of the port to be moved.
`-placement_status` *string*	Specifies the placement status of the port. You can specify: `fixed`—port cannot be moved by automatic tools `placed`—port can be moved by automatic tools

Examples

- The following command moves port vdd2 up over 200 DB units in the vertical direction:

  ```
  move_port -dy 200 pllrst
  ```

Related Information

Affects these commands: syn_generic -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

move_region

```
move_region region
     [-x integer | -dx integer]
     [-y integer | -dy integer]
```

Moves the specified region. You can specify either absolute coordinates, or a delta change in the coordinates. Coordinates or changes in coordinates must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-dx` *`integer`*	Specifies to move the region over the specified distance in the x-direction.
`-dy` *`integer`*	Specifies to move the region over the specified distance in the y-direction.
`-x` *`integer`*	Specifies the new x-coordinate for the lower left corner of the region.
`-y` *`integer`*	Specifies the new y-coordinate for the lower left corner of the region.
`region`	Specifies the name of the region to be moved.

Examples

- The following command moves region `my_reg` over 20 DB units in the vertical direction:

  ```
  move_region -dy 20 [find . -region my_reg]
  ```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_region on page 778
	resize_region on page 826

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

place_dft

`place_dft` *`design`*

Helps to place inserted unplaced logic without disturbing global placement. Useful in cases that after a full physical flow has been run, additional netlist altering commands are also run, resulting in small amounts of mapped but unplaced logic.

Caution

While this command is usually used to place unplaced DFT logic in the design, it will place any unplaced logic. Also, due to the incremental nature of the command, good results will only appear if the percentage of unplaced logic is relatively low. If a big part of the design is unplaced please run a global placement.

Options and Arguments

`design`	Specifies the design in which the DFT logic is to be placed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

place_dft_sequentials

```
place_dft_sequentials
     [-unplaced_insts instances]
     [-list_failed] [-max_print integer]
     [-preview]
```

Places wrapper cells and test point flops.

Use this command to place wrapper cells and test points inserted after `syn_map` was run.

Note: This command requires a `Modus_DFT_Opt` license.

Options and Arguments

`-list_failed`
: Prints the instances that could not be placed.

`-max_print integer`
: Specifies the maximum number of instances for which to report the placement location.

  Default: `20`

`-preview`
: Reports the instances that would be placed with their location without placing them.

`-unplaced_insts instances`
: Specifies the list of DFT instances to be placed.

  By default, the tool attempts to place all unplaced flops and latches.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

predict_floorplan

```
predict_floorplan
     [-constraints file] [-script string] [design]
```

Creates a floorplan by running Innovus Master Plan.

This command invokes Innovus to create a floorplan. By default it will create a square floorplan with an utilization target. You can use option -`constraints` and -`script` to specify constraints/settings to Innovus.

It is recommended to set `innovus_executable` before executing this command.

Note: This floorplan is just a rough starting point and ultimately the user should carefully create a floorplan based on the needs of the design. Things like floorplan size, port locations, macro placement, obstructions, regions can have a very significant impact on the timing, congestion and overall QOR. Hence all these should be defined carefully in a floorplan created by the user. Genus created floorplan is a very preliminary estimate of design QOR while running a Spatial/Physical flow.

Options and Arguments

`-constraints` *`file`*	Specifies the constraints file for Innovus planDesign.
`design`	Specifies the design for which to create the floorplan.
`-script` *`file`*	Specifies the script file for Master plan.

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -spatial
	syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_clip

```
read_clip
     [-keep_existing] [-no_def] [-no_constraints]
     [-input_scale float] [-output_scale float]
     -directory string  [design]
```

Reads the clipper data for the specified design.

Clipper data consists of

- Timing constraints for the clip in Genus Tcl format
- Physical information for the clip in DEF format
- A (binary) database representing the results of the clip operation

Note: If you would read the clip information by sourcing the constraints and using the read_def command, error messages could be generated if pins have changed (if you updated the RTL), By using the `read_clip` command, the messages about different pins are captured and organized.

Options and Arguments

design	Specifies the name of the design. If the design is not specified, Genus will search for a unique design and apply the clipper data to that design. This parameter is only required if multiple designs are loaded.
`-directory` *string*	Specifies the location of the clip data.
`-input_scale` *float*	Multiplies all input constraints by this value. This option allows you to scale input arrival delays. *Default:* `1.0`
`-keep_existing`	Retains constraints currently applied to the design. By default, all constraints are removed prior to reading the clip data. For example, if you applied a clock, using this option will prevent the clip data from overwriting that clock.
`-no_constraints`	Prevents reading the constraints file. Use this option when only the DEF data is desired.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-no_def`
Prevents reading the DEF file. Use this option when only logical analysis is needed.

`-output_scale` *`float`*
Multiplies all output constraints by this value. This option allows you to scale output required times.

Default: `1.0`

Related Information

Related command: create_clip on page 771

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_def

```
read_def
     [-design design]
     [-create_ports] [-power_switch_insert]
     [-no_nets] [-no_specialnets]
     [-no_physical_pins] [-no_vias]
     [-keep_filler_cells] [-keep_welltap_cells]
     [-keep_all_instances} [-keep_all_physical_cells][-keep_routed_nets]
     [-hierarchical [-no_force_fixed] ] [-fuzzy_match]
     [-convert_sites {old_site_name new_site_name}
     [-incremental] [-ignore_errors]
     def_file
```

Loads the specified DEF file.

Genus will perform a consistency check between the DEF and the Verilog netlist and issue relevant messages if necessary. The DEF file must define the die size. A warning message will be issued for any components that lie outside the die area. For better synthesis results, you should also have the pin and macro locations specified in the DEF, although it is not required.

Genus supports DEF 5.3 and above.

Important

The information extracted from the DEF file depends on the license you use to start the tool. In most cases, the `read_def` command will only extract the floorplan information (fixed macros, blockages, pins, regions, and so on). Detailed placement information (in particular, placed components) can only be extracted if you start the tool with a physical license.

Options and Arguments

`-create_ports`	Specifies to create an unconnected top-level netlist port for each pin in the DEF file that does not exist in the netlist and that is either of type clock, reset, scan, or signal.
`-convert_sites {old_site_name new_site_name}`	Converts the row site name specified in the DEF file to the specified new row site name.
`def_file`	Specifies the DEF file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-design` *`design`* Specifies the design to which to annotate the DEF information.

`-fuzzy_match` Enables fuzzy name matching.

`-hierarchical` Specifies that the DEF file is hierarchical.

`-ignore_errors` Ignores any inconsistency errors between the LEF and DEF.

`-incremental` Specifies that the DEF file has incremental information. Therefore only updates of the blockages, components, groups, pins, and regions are needed.

- If a pin or component did not have physical data in the original DEF file, physical data in the incremental DEF are used.
- If a pin or component has physical data from the original DEF file, the physical data in the incremental DEF file will overwrite the original data.

`-keep_all_instances`

Stores all cells instantiated in the `DEF COMPONENTS` section but not instantiated in the netlist as physical-only cells (pcells).

`-keep_all_physical_cells`

Specifies to keep cells with `lib_cell` attribute `keep_as_physical` set to `true`.

`-keep_filler_cells`

Specifies to keep the physical-only filler cells. Filler or spacer cells are standard cells to fill in space between regular core cells.

`-keep_routed_nets` Keeps routed nets in addition to fixed and cover nets.

`-keep_welltap_cells`

Specifies to keep the physical-only welltap cells. Welltap cells are standard cells that connect N and P diffusion wells to the correct power and ground wire.

`-no_force_fixed` Prevents that components have their `placement_status` set to fixed in hierarchical mode (that is, when reading in a hierarchical DEF file).

`-no_nets` Specifies to skip the `NETS` section in the DEF file.

`-no_physical_pins` Specifies not to create a physical pin for a missing pin in the DEF file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-no_specialnets` Specifies to skip the `SPECIALNETS` section in the DEF file.

`-no_vias` Specifies to skip the `VIAS` section in the DEF file.

`-power_switch_insert`

Inserts power switch cells in the netlist if they are present in the DEF file.

These components will be added to the existing netlist and will be connected.

Example

- The following example loads the `point.def` DEF file:

```
legacy_genus:/> read_def point.def
Reading and processing DEF file 'point.def'...
Parsing DEF file...
Warning : Metal fill present. [PHYS-178]
        : Metal fill on layer 'METAL1'.
...
Warning : Component not present in the netlist. [PHYS-171]
        : The component 'U1/foo1' does not exist.
...
Info    : COVER component present. [PHYS-182]
        : The instance 'ram_bank1' is COVER.
Info    : COVER component present. [PHYS-182]
        : The instance 'U1/g20' is COVER.
...
Warning : Physical cell not created due to missing macro. [PHYS-211]
        : Macro 'FILL1MTR' for physical cell 'FILLER2' not found.
Done parsing DEF file.
Processing DEF file...
Warning : Overlapping guide detected. [PHYS-187]
        : Guide 'region_6' (DEF name: 'region_6') overlaps fence
'region_2' (DEF name: 'REGION_TWO').
Warning : Overlapping guide detected. [PHYS-187]
        : Guide 'region_8' (DEF name: 'region_8') overlaps region
'region_1' (DEF name: 'REGION_ONE').
Installing blockage router...
  Summary report for DEF file 'point.def'

    Components
    ----------
        Cover:   3
        Fixed:  10
     Physical:   2
         Bump:   0
       Placed: 591
     Unplaced:   2
        TOTAL: 608 (3 are class macro)

  There are 2 components that do not exist in the netlist.

          Pins
          ----
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
          Cover: 0
          Fixed: 0
       Physical: 0
         Placed: 0
       Unplaced: 0
          TOTAL: 0

            Nets
            ----
           Read: 0 (cover: 0, fixed: 0)
        Skipped: 0
          TOTAL: 0

     SpecialNets
     -----------
           Read: 0
        Skipped: 0
          TOTAL: 0

         Fences: 2
         Guides: 5
        Regions: 1
Done processing DEF file.

  ========================
  Physical Message Summary
  ========================

   2 / 2  W  PHYS-171  Component not present in netlist.
   2 / 2  W  PHYS-178  Metal fill present.
  13 / 0  I  PHYS-181  Full preserve set on instance.
   3 / 3  I  PHYS-182  Cover component present.
   2 / 2  W  PHYS-187  Overlapping guide detected.
   2 / 2  W  PHYS-211  Physical cell not created due to missing macro.
   2 / 2  W  PHYS-214  Library cell not defined in physical library.

Done reading and processing DEF file (time: 0s).
```

Related Information

Affected by this attribute:	script_search_path
Related attribute:	phys_ignore_special_nets

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_encounter

`read_encounter config` *`configuration_file`*

Reads an Innovus configuration file into Genus. An Innovus configuration file is an ASCII file that contains Tcl variables that describe information such as the netlist or RTL, technology libraries, LEF information, constraints, and capacitance tables. Innovus configuration files have the `.config` extension.

After the file is loaded, the constraints and attributes specified in the configuration file will automatically be set. Hence, the design will be ready for synthesis or optimization or both.

Options and Arguments

`configuration_file` Specifies the configuration file to load.

Examples

- Since the configuration file contains information such as technology libraries, HDL files, and constraints, the `read_encounter` command should be used at the beginning of a synthesis session. After the configuration file is loaded, you can immediately synthesize or optimize the design. The following example loads the `fast.config` configuration file, then synthesizes the design to gates.

```
legacy_genus:/> read_encounter config fast.config
legacy_genus:/> synthesize -to_mapped
...
```

Related Information

Related command: write_encounter on page 393

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_physical

`read_physical [-lefs string | -add_lefs string | -oa_ref_libs string]`

Reads the physical design information from the specified LEF files or open access library files. When this command is run, the physical library information is loaded into the database. Both LEF and OA are supported, but not simultaneously.

The `read_physical` command corresponds to the Physical step of the Initialization flow. During this step, the physical libraries (OA or LEF) are loaded into the tool's database using `read_physical` command. When the command is issued, the database is loaded with the specified objects, and is available for querying of attributes associated with the physical data, such as size, layer details, and so on.

The tool uses the standard OA parser, and performs full error checking on the physical data. When the command is complete, a full timing library set and a physical library set would have been loaded into the database.

If -`lefs` option is used, the technology LEF must be first specified.

This command is optional for Synthesis and STA, but is required for Implementation.

Note: This command should be issued after `read_libs` otherwise an error will be issued.

Options and Arguments

`-add_lefs` *`string`*	Specifies a list of LEF files to read and to add to the original list of LEF files that were added before `init_design`, and cannot be combined with any other option. You can add new macros, non-default rules, sites etc. using this option, but you cannot use it to modify any existing LEF data that is already loaded. The LEF files added using this option will be used when the saved design is loaded again using `read_db`.
`-lefs` *`string`*	Specifies the name of the LEF files to read. Specifies the list of LEF files to be loaded. The first LEF file must have the technology section present. The file names are saved to the `init_lef_files` root attribute. This option cannot be combined with -`oa_ref_libs` option.

`-oa_ref_libs` *`string`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the name of the open access library files to read.

Specifies th list of OpenAccess library names to be loaded. These names must also be listed in the cds.lib file. All the abstract views in the specified libraries are loaded.

Use this parameter to specify standard cell libraries or I/O driver libraries that should all be loaded, even if some of the cells are not currently used in the netlist. The first OpenAccess library that is specified is also used to load the OpenAccess technology data.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_sdp_file

```
read_sdp_file -file file
     [-design design] [-preserve_size_ok]
     [-hier_path path] [-origin llx lly]
```

Reads a relative placement file (in .sdp format) containing the SDP definitions.

Options and Arguments

`-design` *`design`*	Specifies the design in which to use the SDP information.
`-file` *`file`*	Specifies the name of the SDP input file.
`-hier_path` *`path`*	Specifies the hierarchical path name of the SDP elements specified in the SDP file. This path name is appended to all SDP instances defined in the file.
`-origin` *`llx lly`*	Specifies the coordinates of the origin of the SDP groups. You can use floats to specify the coordinates. All SDP groups use the same origin.
`-preserve_size_ok`	Sets the `preserve` attribute on the SDP instances to `preserve_size_ok`. *Default:* `preserve`

Example

The following command reads the `dtmf_chip.sdp` file. It also appends the specified hierarchical path to all elements.

```
read_sdp_file -file dtmf_chip.sdp -hierPath DTMF_CHIP/TDSP_CORE_INST
```

Related Information

SDP Flows in *Genus Physical Guide for Legacy UI*

Related command: write_sdp_file on page 840

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_spef

```
read_spef spef_file
     [-max_fanout integer] [-rc_corner rc_corner]
     [-hierarchical] [-incremental]
```

Reads the SPEF file and loads the resistance and grounded capacitors from the file. Gzip compressed files (`.gz` extension) can also be loaded. In Genus, the SPEF file is generated by Innovus.

Options and Arguments

`-hierarchical`	Specifies that SPEF file is hierarchical.
`-incremental`	Specifies that the SPEF file contains incremental information. Allows to read in the data without resetting the original SPEF annotated values (if any).
`-max_fanout`	Any net with a fanout count greater than the specified value will not be annotated with the resistance and capacitance from the SPEF. Also, delay calculation will not be performed. The default value is1000.
`-rc_corner` *`rc_corner`*	Specifies the RC corner object that the SPEF values are associated with. **Note:** This option is not supported in the legacy UI. If you specify this option, the tool will return an error message.
`spef_file`	Specifies the SPEF file.

Related Information

Related command: write_spef on page 841

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report congestion

Refer to report congestion in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report utilization

Refer to report utilization in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

resize_blockage

```
resize_blockage blockage
     [-left integer] [-right integer]
     [-top integer] [-bottom integer]
```

Resizes the specified blockage. All distances must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-bottom` *`integer`*	Specifies the distance over which the bottom edge of the blockage must be moved.
`-left` *`integer`*	Specifies the distance over which the left edge of the blockage must be moved.
`-right` *`integer`*	Specifies the distance over which the right edge of the blockage must be moved.
`-top` *`integer`*	Specifies the distance over which the top edge of the blockage must be moved.
`blockage`	Specifies the name of the blockage to be resized.

Example

The following command extends blockage `my_blck` with 20DB units on the right side.

```
resize_blockage my_blck -right 20
```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_placement_blockage on page 775
	create_placement_halo_blockage on page 777
	move_blockage on page 803

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

resize_region

```
resize_region region
     [-left integer] [-right integer]
     [-top integer] [-bottom integer]
```

Resizes the specified region. All distances must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

Options and Arguments

`-bottom` *`integer`*	Specifies the distance over which the bottom edge of the region must be moved.
`-left` *`integer`*	Specifies the distance over which the left edge of the region must be moved.
`-right` *`integer`*	Specifies the distance over which the right edge of the region must be moved.
`-top` *`integer`*	Specifies the distance over which the top edge of the region must be moved.
`region`	Specifies the name of the region to be resized.

Example

The following command extends region `my_reg` with 20DB units on the right side.

```
resize_region my_reg -right 20
```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical
Related commands:	create_region on page 778
	move_region on page 808

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

restore_congestion_map

`restore_congestion_map [-design design] file`

Restores the congestion map data from the specified file.

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`-design design`	Specifies the design for which the congestion map must be restored.
`file`	Specifies the name of the file containing the congestion data.

Related Information

Related commands: write_congestion_map on page 836
update_congestion_map on page 833

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

specify_cell_pad

```
specify_cell_pad
     -padding string
     libcell_list
```

Note: Specifies the padding factor to be used to calculate the padding (placement clearance) for instances of the specified library cell types.

During placement, the tool adds the padding to the right side of the instances of the specified cell types. The padding is determined by multiplying the placement SITE value of the libcells with the specified padding factor.

Note: The padding is retained during placement optimization and clock-tree synthesis.

Options and Arguments

libcell_list	Specifies a list of libcell types to which the padding factor applies.
-padding *string*	Specifies the padding factor to be applied.

Example

The following command specifies to add a padding of two times the placement site value of libcell `inv`.

```
specify_cell_pad -padding 2.0 /libraries/slow/libcells/inv
```

Related Information

Affects these commands:	syn_generic -physical
	syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

specify_floorplan

```
specify_floorplan
     { -die_box {llx lly urx ury}
     | -die_points pt pt pt [pt]...
     | -height float -width float }
     [-core_box {llx lly urx ury} ]
     {design | subdesign}
```

Specifies the floorplan information for the specified design.

Options and Arguments

`-core_box {llx lly urx ury}`

Specifies the lower left and upper right coordinates of the core of the design. Specify the coordinates in micron.

`{design | subdesign}`

Specifies the name of the design or subdesign.

`-die_box {llx lly urx ury}`

Specifies the lower left and upper right coordinates of the die of the design. Specify the coordinates in micron.

`-die_points pt pt pt [pt]...`

Specifies a list of coordinates of at least three points if the die has a polygon geometry. Specify the coordinates in micron.

`-height float` Specifies the height of the die. Specify the height in micron.

`-width float` Specifies the width of the die. Specify the width in micron.

Examples

- The following commands first specify the width and height of the die, then the coordinates of the core. Parameter legality checking is performed as shown below.

```
legacy_genus:/> specify_floorplan DTMF_CHIP -height 1400 -width 1200
legacy_genus:/> specify_floorplan DTMF_CHIP -core_box {100 100 1400 1400}
Error   : The design core box must lie within the die box. [PHYS-102]
[specify_floorplan]
        : core box = {100 100 1400 1400}, die box = {0 0 1200 1400}
        : Wrong coordinates were specified for the core box.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command specifies the coordinates of the die and the core of the design.

```
specify_floorplan MYCHIP -die_box {0 0 1550 1500} -core_box {100 100 1400 1400}
```

Related Information

Affects these commands: syn_generic -physical
syn_opt -physical

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

summary_table

`summary_table -outdir path`

Generates a summary table which includes various QoS numbers for various stages in the RC flow.

Note: You must have run the `generate_reports` before you use this command.

Options and Arguments

`-outdir path`	Specifies the path to the directory where the output data of the this command must be stored.

Example

The following report shows the summary of the statistics after three stages: `initial` stage, `generic` stage, and `map` stage, where `initial`, `generic`, and `map` were the tags specified in the subsequent `generate_reports` commands.

```
legacy_genus:/> summary_table -outdir $env(TEST_OUT_DIR)


Working Directory = /...test
QoS Summary for cscan
================================================================================
Metric                   initial    generic    map
================================================================================
Slack (ps):                1958.6     1934.1     1190.1
  R2R (ps):              no_value   no_value   no_value
  I2R (ps):                1958.6     1934.1     1190.1
  R2O (ps):                2152.6     2152.6     1987.2
  I2O (ps):              no_value   no_value   no_value
  CG  (ps):                2100.0     2100.0     2100.0
TNS (ps):                       0          0          0
  R2R (ps):              no_value   no_value   no_value
  I2R (ps):                     0          0          0
  R2O (ps):                     0          0          0
  I2O (ps):              no_value   no_value   no_value
  CG  (ps):                     0          0          0
Failing Paths:                  0          0          0
Area:                           0          0          0
Instances:                     36         28         43
Utilization (%):             0.00       0.00       0.00
Tot. Net Length (um):  no_value   no_value   no_value
Avg. Net Length (um):  no_value   no_value   no_value
Total Overflow H:               0          0          0
Total Overflow V:               0          0          0
Route Overflow H (%):  no_value   no_value   no_value
Route Overflow V (%):  no_value   no_value   no_value
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
================================================================================
CPU  Runtime (m:s):        00:04      00:00      00:01
Real Runtime (m:s):        00:13      01:35      01:30
CPU  Elapsed (m:s):        00:09      00:10      00:12
Real Elapsed (m:s):        00:37      02:12      03:42
Memory (MB):               66.32      66.60      73.38
=============================================================================

================================================================================
Flow Settings:
================================================================================
Total Runtime (m:s): 03:25
Total Memory (MB):   73.38
Executable Version:  9.1
================================================================================
```

Related Information

Affected by this command:	generate_reports on page 797
Affected by this attribute:	qos_report_power

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

update_congestion_map

`update_congestion_map [-pin_density] [-design design]`

Updates the congestion map for the specified design.You can run this command after any command that updates the physical data.

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`-design design`	Specifies the design for which the congestion map must be updated.
`-pin_density`	Specifies to compute the pin density.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

update_group

```
update_group -name group]
     {-add | -delete} [-objs string]
```

Adds or removes the specified objects from the specified group

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`-add	-delete`	Adds or removes the specified objects.
`-name` *`group`*	Specifies the name of the group for which to ad or remove instances.
`-objs` *`string`*	Specifies the instances, hierarchical instances or groups to be added or removed to the specified disjointedness.

Related Information

Related command: create_group on page 772

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

update_instance_data

```
update_instance_data
     [-dbu] [-location_x float] [-location_y float]
     [-placement_status string] [-orientation string]
     {instance}
```

Updates the location and placement status of the specified instance.

Note: This command applies only to the Genus-Physical flow.

Options and Arguments

`-dbu`	Specifies to use DB Units for the coordinates.
`{instance}`	Specifies the name of the instance for which the data needs to be updated.
`-location_x float`	Specifies the lower left corner x-coordinate.
`-location_y float`	Specifies the lower left corner y-coordinate.
`-orientation`	Specifies the orientation.
`-placemet_status string`	Specifies the instances, hierarchical instances or groups to be added or removed to the specified group.

Related Information

Related command: create_group on page 772

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_congestion_map

`write_congestion_map [-design design] file`

Saves the congestion map to the specified file.

Note: This command applies only to the Genus-Physical flow.

This command is also aliased to `save_congestion_map`.

Options and Arguments

`-design` *`design`*	Specifies the design for which the congestion map must be saved.
`file`	Specifies the name of the file to which the congestion data must be saved.

Related Information

Related commands: restore_congestion_map on page 827
update_congestion_map on page 833

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_def

```
write_def [-ignore_groups] [-ignore_placed_instances]
     [-hierarchical instance] [-scan_chains]
     [-no_special_net][-no_def_pin]
     [-version string] [design] [> file]
```

Writes floorplan information to a DEF file. The DEF, by default will not contain netlist information (net connectivity information) aside from the power/ground nets defined in the input DEF (`SPECIALNETS` section).

You can write out the DEF in gzip format by specifying the `.gz` extension when writing out the file.

You can use the `write_def` command after importing a design and reading in a floorplan DEF or creating a floorplan.

To create a DEF in gzip format specify .gz extension when writing out the file.

Note: You can select the DEF version for your output file by setting the `def_output_version` attribute to the version number. For example, to set the version number to 5.6, type the following command:

```
set_attr def_output_version 5.6
```

Options and Arguments

design	Specifies a particular design for which to write out the floorplan. Only one design can be specified at a time.
file	Redirects the floorplan to the specified file.
-hierarchical *instance*	Specifies the hierarchical instance for which to write out the DEF.
-ignore_groups	Discards the instance groups that are defined in RC. These come from the input DEF (`GROUPS` section). The output DEF will have no `GROUPS` or `REGIONS` sections.
-ignore_placed_instances	Only writes preplaced instances (`+ FIXED` tag in the DEF) to the DEF. These are objects such as macros. Without this option, the output DEF will include all the instances that are preplaced or placed. Unplaced components will never be written to the DEF. This is all in the `COMPONENTS` section.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-no_def_pin`	Specifies to exclude `def_pin` objects from the written out DEF file.
`-no_special_net`	Specifies to exclude scpecial net information from the written out DEF file.
`-scan_chains`	Specifies that scan chain information should be included in the floorplan.
`-version` *`string`*	Specifies the DEF version to write out. *Default:* `5.7`

Related Information

Related attribute: phys_ignore_special_nets

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_lib_lef

```
write_lib_lef
     [-lib file] [-lef file]
     [-process_node integer] [design]
```

Writes out the timing model for the design in Liberty format and/or the physical model in LEF library format.

Options and Arguments

design	Specifies the design for which to write out the library data.
-lef *file*	Specifies the name of the file to which to write out the LEF data stored in the database.
-lib *file*	Specifies the name of the file to which to write out the Liberty data stored in the database.
-process_node *integer*	Specifies the process design node for which to write out the information.

Examples

- Simple usage example:

  ```
  write_lib_lef -lib top -lef top
  ```

  Output files:

  ```
  Timing model: top.lib
  Physical model: top.lef
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_sdp_file

```
write_sdp_file
     [-file file]
     [-top_group sdp_group]
```

Saves the current SDP relative placement information in the specified file. Genus stores the original design data along with additional information in the SDP file in the design information hierarchy in the form of attributes.

Options and Arguments

`-file file` Specifies the name of the SDP output file.

Default: Output is written to the screen.

`-top_group sdp_group`

Specifies the SDP top group whose information to write out.

Example

The following command writes the current SDP information in SDP format in file `test.sdp`.

```
write_sdp_file  -file test
```

Related Information

SDP Flows in *Genus Physical Guide for Legacy UI*

Related command:	read_sdp_file on page 821
Related attributes:	sdp_row
	sdp_column
	sdp_instances
	sdp_group

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_spef

```
write_spef
     [-cap_unit {pf | fF}]
     [-res_unit {ohm | kohm}]
     [> file]
```

Writes out the parasitics (resistance and capacitance information) of the design in SPEF (Standard Parasitic Exchange Format) format.

Options and Arguments

`-cap_unit {pf | fF}`

Specifies the unit to be used to write out the capacitance values.

Default: `pf`

`file`

Specifies the file to which to write the parasitics.

`-res_unit {ohm | kohm}`

Specifies the unit to be used to write out the resistance values.

Default: `ohm`

Related Information

Related command: read_spef on page 822

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

13

Design for Test

- add_analyzed_test_points on page 848
- add_opcg_hold_mux on page 851
- analyze_scan_compressibility on page 854
- analyze_testability on page 864
- check_atpg_rules on page 867
- check_dft_pad_configuration on page 869
- check_dft_rules on page 870
- check_dft_setup on page 877
- compress_block_level_chains on page 878
- compress_scan_chains on page 882
- concat_scan_chains on page 897
- configure_pad_dft on page 899
- connect_cgic_test_pins_controlling_wrapper_cells on page 900
- connect_compression_clocks on page 901
- connect_dft_top_level on page 903
- connect_opcg_segments on page 904
- connect_scan_chains on page 906
- create_chains_database on page 911
- define_dft on page 912
- define_dft abstract_segment on page 915
- define_dft boundary_scan_segment on page 921

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- define_dft dft_configuration_mode on page 925
- define_dft domain_macro_parameters on page 927
- define_dft fixed_segment on page 929
- define_dft floating_segment on page 931
- define_dft jtag_instruction on page 933
- define_dft jtag_instruction_register on page 937
- define_dft jtag_macro on page 939
- define_dft mbist_clock on page 944
- define_dft opcg_domain on page 947
- define_dft opcg_mode on page 951
- define_dft opcg_trigger on page 953
- define_dft osc_source on page 956
- define_dft pmbist_direct_access on page 958
- define_dft pmbist_port on page 962
- define_dft preserved_segment on page 963
- define_dft scan_chain on page 966
- define_dft scan_clock_a on page 972
- define_dft scan_clock_b on page 975
- define_dft shift_enable on page 978
- define_dft shift_register_segment on page 981
- define_dft tap_port on page 983
- define_dft test_bus_port on page 985
- define_dft test_clock on page 990
- define_dft test_mode on page 995
- define_test_signal on page 1000
- dft_trace_back on page 1008
- fix_dft_violations on page 1010

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- fix_scan_path_inversions on page 1014
- identify_domain_crossing_pins_for_cgic_and_scan_abstracts on page 1015
- identify_multibit_cell_abstract_scan_segments on page 1016
- identify_shared_wrapper_cells_in_design on page 1018
- identify_shift_register_scan_segments on page 1019
- identify_test_mode_registers on page 1021
- insert_dft on page 1024
- insert_dft boundary_scan on page 1026
- insert_dft compression_logic on page 1031
- insert_dft hard_repair on page 1042
- insert_dft jtag_macro on page 1046
- insert_dft lockup_element on page 1050
- insert_dft logic_bist on page 1051
- insert_dft opcg on page 1063
- insert_dft pmbist on page 1065
- insert_dft ptam on page 1071
- insert_dft scan_power_gating on page 1074
- insert_dft shadow_logic on page 1077
- insert_dft shift_register_test_points on page 1082
- insert_dft test_point on page 1083
- insert_dft user_test_point on page 1089
- insert_dft wir_signal_bits on page 1092
- insert_dft wrapper_cell on page 1094
- insert_dft wrapper_instruction_register on page 1102
- insert_dft wrapper_mode_decode_block on page 1104
- insert_test_compression on page 1106
- map_dft_unmapped_logic on page 1109

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- map_pmbist_cgc_to_cgic on page 1110
- map_pmbist_ffn on page 1111
- map_pmbist_ffsync on page 1112
- read_dft_abstract_model on page 1113
- read_io_speclist on page 1116
- read_memory_view on page 1117
- read_pmbist_interface_files on page 1120
- replace_opcg_scan on page 1121
- replace_scan on page 1123
- report dft_chains on page 1125
- report dft_clock_domain_info on page 1126
- report dft_core_wrapper on page 1127
- report dft_registers on page 1128
- report dft_setup on page 1129
- report dft_violations on page 1130
- report opcg_equivalents on page 1131
- report scan_compressibility on page 1132
- report test_power on page 1133
- report_chains_longest_wires on page 1134
- report_chains_total_wirelength on page 1135
- report_chains_wires on page 1136
- report_scan_compression_logic on page 1137
- reset_opcg_equivalent on page 1138
- reset_scan_equivalent on page 1139
- reset_scan_equivalent on page 1139
- set_compatible_test_clocks on page 1140
- set_opcg_equivalent on page 1142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- set_scan_equivalent on page 1144
- update_scan_chains on page 1146
- write_atpg on page 1148
- write_bsdl on page 1151
- write_compression_macro on page 1154
- write_dft_abstract_model on page 1162
- write_dft_compression_test_points on page 1165
- write_dft_deterministic_test_points on page 1169
- write_dft_lbist_test_points on page 1173
- write_dft_opcg_domain_macro_rtl on page 1177
- write_dft_opcg_trigger_macro_rtl on page 1178
- write_dft_rtl_model on page 1179
- write_et_atpg on page 1181
- write_et_bsv on page 1188
- write_et_lbist on page 1192
- write_et_no_tp_file on page 1197
- write_io_speclist on page 1199
- write_logic_bist_macro on page 1201
- write_memory_physical_map on page 1206
- write_pmbist_interface_files on page 1207
- write_pmbist_testbench on page 1210
- write_scandef on page 1216

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_analyzed_test_points

```
add_analyzed_test_points
     -input_test_point_file file
     [-max_number_of_test_points integer ]
     [-min_slack integer]
     [-fault_threshold integer]
     [-share_observe_flop integer]
     [-control_only] [-observe_only]
     [-test_clock_pin {port|pin}]
     [-test_control test_signal]
     [-gate_clock_test_control test_signal]
     [-gate_clock_for_observe test_signal]
     [-dont_map] [-verbose] [design]
```

Inserts selected test points analyzed by Modus Compression, LBIST, or Deterministic Testpoint Analysis commands. Test points can be inserted that have minimal impact on the slack and which target a minimum specified fault count.

Options and Arguments

`-control_only`
: Specifies to insert only control test points.

design
: Specifies the name of the top-level design on which you want to perform test-point insertion.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dont_map`
: Prevents the inserted test points from being mapped even if the design is already mapped to the target library.

`-fault_threshold` *integer*
: Specifies to insert only those test point locations whose fault count is greater than or equal to the number specified.

  Default: 0

`-gate_clock_for_observe` *test_signal*
: Specifies the test signal to be used to gate the clock of the observation flop.

`-gate_clock_test_control` *test_signal*
: Specifies the test signal to be used to control the gating logic of the clock of the test point.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-input_test_point_file` *`file`*

Specifies the name of the file containing the test point locations.

The file is specified in Modus format.

`-max_number_of_test_points` *`integer`*

Specifies the number of test points to be inserted.

You must specify an integer value greater than `0` when attempting to insert test points from a file using the `-input_test_point_file` option.

By default, all test points are inserted.

`-min_slack` *`integer`* Limits the insertion of a test point to those nodes that have the specified minimum slack (in ps).

Default: `-10000000`

`-observe_only` Specifies to insert only observation test points. In this case, you do not need to specify a test control signal.

`-share_observe_flop` *`integer`*

Specifies the number of observation test nodes that can share an observation flop through an XOR tree.

Default: `1`

`-test_clock_pin {`*`port`*`|`*`pin`*`}`

Specifies the test clock that drives the clock pin of the inserted test points during test mode operation. Specify a port or pin that drives the test clock.

If this option is not specified, the tool uses the test clock pin of the first flop in the fanin cone. If the tool cannot find any test clock pin in the fanin cone, it uses the first test clock in the `dft/test_clock_domains` directory.

`-test_control` *`test_signal`*

Specifies the test signal to use to control the test points.

Note: You must have specified the test signal using the `define_dft test_mode` constraint.

`-verbose` Specifies to print test point details.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command inserts all of the test points, and when possible 3 test point locations will be shared through an XOR-tree to a common observation flop.

```
add_analyzed_test_points \
-input_test_point_file rc_et_dfa/myDesign.dfa.tpfile \
-test_control test_mode1 \
-test_clock_pin clk \
-share_observe_flop 3
```

- The following example inserts the first 500 test points whose fault count is greater than or equal to 3 from the specified test point file `myfile`.

```
add_analyzed_test_points -max_number_of_test_points 500 \
 -fault_threshold 3 -test_control tm -test_clock_pin clk \
-input_test_point_file myfile
```

- The following example inserts all test points for test nodes with a minimum slack of 3ps, and will share 3 observation test nodes through an XOR tree from the specified test point file `myfile`.

```
add_analyzed_test_points -min_slack 3 -share_observe_flop 3 \
-test_control tm -test_clock_pin clk -input_test_point_file myfile
```

- The following command inserts a maximum of 10 DFA test points whose nodes have a minimum slack of 1000 ps. Also, when possible, three test point locations will be shared through an XOR-tree to a common observation flop.

```
add_analyzed_test_points \
  -input_test_point_file rc_et_dfa/myDesign.dfa.tpfile \
  -min_slack 1000 \
  -test_control test_mode1 \
  -test_clock_pin clk \
  -share_observation_flop 3 \
  -max_number_of_test_points 10
```

- The following command inserts a maximum of 10 test points for locations whose fault count is greater than or equal to 15:

```
add_analyzed_test_points \
  -input_test_point_file rc_et_dfa/myDesign.dfa.tpfile \
  -test_control test_mode1 \
  -test_clock_pin clk \
  -max_number_of_test_points 10
  -fault_threshold 15
```

Related Information

Affected by these constraints: define_dft test_clock on page 990

define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_opcg_hold_mux

```
add_opcg_hold_mux
     -edge_mode test_signal
     -instance instance
     [design]
```

Replaces the specified domain blocking scan flop instance with its `OPCG`-equivalent if you specified `OPCG`-equivalence mappings with the `set_opcg_equivalent` command. If the `OPCG`-equivalence mappings are not available, or if the scan cell type of the instance does not have an entry in the `OPCG`-equivalence table, the command adds a hold mux before the scan cell to convert it to an `OPCG`-equivalent cell and makes all necessary connections.

This command is used for designs with OPCG logic when the OPCG domain blocking attribute is enabled in the flow. Domain crossing flops are identified when the scan chains are built. A domain crossing flop will have its instance level attribute '`dft_opcg_domain_crossing flop`' set to `true`.

Options and Arguments

design
Specifies the design in which you want to replace an instance.

`-edge_mode` *test_signal*
Specifies the global edge-mode signal to connect.

`-instance` *instance*
Specifies the instance to be replaced.

Related Information

Affected by these commands: reset_opcg_equivalent on page 1138
set_opcg_equivalent on page 1142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_pmbist_access_method

```
add_pmbist_access_method -asu_vectors_file string
     [-module_prefix string] [-alt_dft_cfg_mode dft_config_mode]
     [-dont_scan] [-directory string] [-fcu_instance hinst]
     [-amu_instance hinst] [design]
```

Adds Programmable Built-In-Self-Test (PMBIST) capabilities to test the memories. It requires a vector file generated by `write_pmbist_testbench` or `create_embedded_test` command. The access method hardware needs to be inserted at block levelafter PMBIST specific hardware insertion before moving on to top-level PMBIST insertion (bottom-up).

Control and monitor signals for power-on-self-test are defined using `define_dft pmbist_direct_access` command.

Useful Signals:

- `mda_run` - enables power-on-self-test
- `mda_run_mask` - to control what needs to be tested
- `mda_run_done`, `mda_run_ovfl` and `mda_run_fail` - to monitor the results of the testing

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-alt_dft_cfg_mode` *`dft_config_mode`*

Specifies the configuration mode in which netlist back-tracing is to be done.

Default: pmbist_od

`-amu_instance` *`hinst`*

Specifies the name of the associated algorithm memory unit instance.

`-asu_vectors_file` *`string`*

Specifies the vector file for application specific unit generation.

`design` Specifies the name of a top-level design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-directory string`
: Specifies the working directory.

  Default: ./pmbist_od_<designname>

`-dont_scan`
: Allows access method insertion flow without test signals when the access method flops are not scan replaced.

`-fcu_instance` *`hinst`*
: Specifies the name of the associated fuse control unit instance.

`-module_prefix` *`string`*
: Specifies an additional character string to append to the default prefix `tem`. The string is placed on all modules created and inserted by this command.

Related Information

Related commands:

define_dft pmbist_direct_access on page 958

write_pmbist_testbench on page 1210

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

analyze_scan_compressibility

```
analyze_scan_compressibility
     -library string [design]
     [-scanins integer...] [-scanouts integer...]
     [-channel_lengths integer...]
     [-min_scanned_flop_percentage integer]
     [-compression_method {add_test_compression | compress_scan_chains}]
     [-compressor {xor | misr | mimic_bidi_misr}]
     [-decompressor {xor | broadcast}]
     [-mask {wide1 | wide0 | wide2 | none}] [-serial_misr_read]
     [-fault_sample_ratio float]
     [-ratio_list list_of_compression_ratios] [-delay]
     [-directory directory] [-dont_run_atpg]
     [-atpg_options string] [-build_model_options string]
     [-build_faultmodel_options string]
     [-build_testmode_options string]
     [-verify_test_structures_options string] [-verbose]
```

Analyzes a design for scan-based compressibility and produces actual compression results for each compression setting. Analysis is done based on the scan chain information in the design.

For the `compress_scan_chains` compression method:

1. If actual scan chains exist, analysis is performed based on the number of actual scan chains.
2. If actual scan chains do not exist, analysis is performed based on the number of user-defined scan chains.
3. If user-defined scan chains do not exist, analysis is performed based on the number of chains specified using the `-scanins` option.

Note: When using the `compress_scan_chains` method, you are expected to provide a list of ratios using the `-ratios` option. When the `-ratios` option is not specified for this compression method, the command runs the analysis using the `-ratios` option with the default values of `"20 50 100"`

For the `add_` compression method:

1. If actual scan chains exist, analysis is performed based on the number of actual scan chains (internal compression chains) and specifying one of the additional options `-scanins` or `-ratios`.

   For each value provided to the `-scanins` option, the configuration is built and the resulting ratio is reported.

   If instead, a list of values is provided to the `-ratios` option, then each configuration is built and the computed number of scan in pins is reported.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

2. If actual scan chains do not exist, analysis is performed based on specifying two of the three options `-scanins`, `-ratios` and `-channel_lengths`. One of these options must be provided as a list of values and a single value must be provided for the other. The third option will be determined automatically from the specified options. For example:

   The tool will determine the `channel_lengths` for

   ```
   -scanins 8 -ratios "5 10 15"
   ```

   The tool will determine the number of `scanin` pins for

   ```
   -channel_lengths 100 -ratios "5 10 15"
   ```

   The tool will calculate the scan compression ratio for

   ```
   -channel_lengths 100 -scanins "5 10 15"
   ```

The command usages above assume the compression analysis is performed for *symmetric* compression, where the number of scan input signals equals the number of scan output signals. Symmetric compression analysis is performed without specifying the number of scan output signals.

Using the `add_test_compression` compression method, you can also perform an *asymmetric* compression analysis by specifying the `-scanouts` option. This analysis is used to verify the impact of aliasing by varying the number of scan-out ports. You must invoke `analyze_scan_compressibility` with `-atpg_options "aliascheck=yes"` for Modus ATPG to enable aliasing checking during pattern generation which can lead to longer ATPG runtimes.

- When compression chains do not exist, the tool determines the `channel_lengths` and will vary the `scanouts`:

  ```
  -scanins 8 -ratios 2 -scanouts "2 4"
  ```

- When compression chains exist, you can specify either scan-ins or ratios and vary the scan-outs

  ```
  -scanins 8 -scanouts "2 4"
  -ratios  2 -scanouts "2 4"
  ```

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-atpg_options` *`string`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies a string containing the extra options to run ATPG-based testability analysis.

Use the following format for the *string*:

"-option1 value -option2 value..."

Note: For more information on these options, refer to the `create_logic_tests` or `create_logic_delay_tests` command in the *Command Line Reference* (of the Modus documentation).

`-build_faultmodel_options` *string*

Specifies a string containing the extra options to build a fault model.

Note: For more information on these options, refer to the `build_faultmodel` command in the *Command Line Reference* (of the Modus documentation).

`-build_model_options` *string*

Specifies extra options to apply when building the model.

Use the following format for the *string*:

"-option1 value -option2 value ..."

Note: For more information on these options, refer to the `build_model` command in the *Command Line Reference* (of the Modus documentation).

`-build_testmode_options` *string*

Specifies extra options to apply when building the test mode.

Use the following format for the *string*:

"-option1 value -option2 value ..."

Note: For more information on these options, refer to the `build_testmode` command in the Modus documentation.

`-channel_lengths` *integer*

Specifies the lengths of the scan compression channels to be built using the `add_test_compression` compression method. This option is only required if no actual scan chains exist.

The specified number must be equal to or larger than one.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-compression_method {compress_scan_chains|add_test_compression}`

Specifies the compression method to use:

- `add_test_compression` adds compression logic for designs starting with scan compression (internal) chains
- `compress_scan_chains` adds compression logic for designs starting with Fullscan chains

Default: `add_test_compression`

`-compressor {xor | misr | mimic_bidi_misr}`

Specifies the type of compression logic for analysis:

- `xor` analyzes an XOR-based compressor
- `misr` analyzes a `MISR-based` compressor using the `compress_scan_chains` compression method
- `mimic_bidi_misr` analyzes an on-product MISR compressor using the `compress_scan_chains` compression method and mimics the behavior of the design as though the pads can be configured bidirectionally.

Default: `xor`

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic for analysis:

- `broadcast` specifies broadcast-based decompression logic (simple scan fanout).
- `xor` specifies an XOR-based spreader network in addition to the broadcast-based decompression logic.

Default: `xor`

`-delay`

Specifies to create the Modus run scripts for Delay Test ATPG. This option must also be used when the design includes OPCG logic.

`design`

Specifies the name of the top-level design on which to perform analysis.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *`atpg_directory`*

Specifies the directory to run and store ATPG results.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: ATPG results tend to consume high amounts of disk space for larger designs.

Default: `current_working_directory/asc`

`-dont_run_atpg`

Specifies to only insert the different compression options and produces a list of the ATPG jobs to be run.

Note: This option can be used in combination with the `report_scan_compressibility` command as the list of ATPG jobs is produced. The ATPG jobs may be run with either of the following methods:

- Run on a different workstation.
- Run in parallel mode to reduce runtime if a Load Sharing Facility (LSF) environment is available.

Important

Neither of the preceding methods are automatically invoked by the `analyze_scan_compressibility` command or the `report_scan_compressibility` command.

`-fault_sample_ratio` *`float`*

Specifies the percentage of faults used to estimate the test coverage.

This option will be ignored when the command is specified with the `-delay` option.

Defaults: `100.0` for full ATPG, valid values between `0.01` and `50.0`

`-library` *`string`*

Specifies the list of Verilog structural library files.

You can specify the files explicitly or you can specify an *include* file that lists the files. You can also specify directories of Verilog files but you cannot reference directories in an include file.

For example, assume the following Verilog files are required:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

You can specify the files in either of the following ways:

1. Explicitly:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
analyze_scan_compressibility -library "./padcells.v \
 ./stdcells.v ./memories ./ip_blocks" ...
```

2. Using an include file.

```
analyze_scan_compressibility \
-library "include_libraries.v ./memories \
 ./ip_blocks" ...
```

where you created a file `include_libraries.v` with the following contents:

```
`include "./padcells.v"
`include "./stdcells.v"
```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-mask {wide1 | wide0 | wide2 | none}`

Specifies the scan channel masking logic type for analysis. The masking types that can be used depend on the compressor type specified with the `-compressor` option.

By default, `wide1` masking logic is inserted.

Genus provides three types of channel masking: `wide0`, `wide1`, and `wide2`.

- For XOR-based compression, two types of masking are supported: `wide1` and `wide2`
- For `MISR`-based compression, three type of masking are supported: `wide0`, `wide1`, and `wide2`.

When requested to insert `wide1` masking, one mask register bit is added per channel. When requested to insert `wide2` masking, two mask register bits are added per channel.

`-minimum_scanned_flop_percentage` *`integer`*

Specifies the minimum percentage of scannable flops required to allow the scan compression analysis to proceed.

Default: `95`

`-ratio_list` *`integer_list`*

Specifies the list of compression ratios to be analyzed.

Default: `"20 50 100"` using `compress_scan_chains` compression method

`-scanins` *`integer`* Specifies the number of top-level scan input pins.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The specified number must be larger than one.

`-scanouts` *`integer`*
: Specifies the number of top-level scan output pins or the list of numbers of top-level scan output pins to be analyzed.

  The specified number must be larger than one.

  Default: Equal to the number of specified, or calculated top-level scan input pins for symmetric compression.

`-serial_misr_read`
: Performs analysis for serial `MISR` compression using `compress_scan_chains` compression method.

  You can only specify this option for a `MISR`-based compression (`-compressor` option is set to `misr)`.

  You cannot specify this option when the `-decompressor` option is set to `xor`.

`-verbose`
: Displays all messages during analysis of the design for scan-based compressibility.

`-verify_test_structures_options` *`string`*
: Specifies extra options to apply when performing test structure verification.

  Use the following format for the *`string`*:

  `"-option1 value -option2 value ..."`

  Note: For more information on these options, refer to the `verify_test_structures` command in the Modus documentation.

Examples

- The following command performs XOR-based compression analysis with `wide1` masking for a ratio list of `"20 50 100"`. The command does not perform the ATPG runs of the different compression schemes because the `-dont_run_atpg` option was specified.

```
legacy_genus:/> analyze_scan_compressibility  -chains 8 -compressor xor -mask \
wide1 -ratio_list "20 50 100" -dont_run_atpg -library mylibrary -directory asc
Starting scan compressibility analysis for module 'test'

Creating work directory asc .....


Analyzing the design with the following configuration
-----------------------------------------------------
        Design       - test
        Decompressor - broadcast
        Compressor   - xor
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
        mask         - wide1
        ratio        - 20 50 100

Scan Registers Status
---------------------
        (Number of Scannable Registers) / (Total Register Count) =
        --> 999 / 999 = 100%

        (Number of flops mapped to scan) = 999
        (Number of flops passing tdrc)   = 999

        (Number of Scannable Registers)
        (Number of flops scan mapped and passing tdrc) +
        (Number of flops in shift-register segments) +
        (Number of flops in abstract scan segments) =
        --> 999 + 0 + 0 = 999

        (Total Register Count)
        (Number of registers) +
        (Number of flops in abstract scan segments) =
        --> 999 + 0 = 999

        (*) The percentage of scannable registers is 100%.

Building the Fullscan Chains
----------------------------

        Building balanced scan chains assuming all test clocks are compatible
        and mixing of clock edges in scan chains is allowed.
        If you prefer a different set of constraints, please build the scan
        chains before calling analyze_scan_compressibility.

        Configuring 8 chains for 999 scan f/f
                ...chain01 of length: 125
                ...chain02 of length: 125
                ...chain03 of length: 125
                ...chain04 of length: 125
                ...chain05 of length: 125
                ...chain06 of length: 125
                ...chain07 of length: 125
                ...chain08 of length: 124

        Modus scripts for ATPG have been written
        for fullscan mode to directory 'asc/et_scripts_fullscan'.
        .... Invoke the script from OS command line as 'et -e
        asc/et_scripts_fullscan/runet.atpg'.

Inserting Scan Compression Logic
--------------------------------

        Compressing scan chains with ratio 20
        Tool achieved ratio: 17.8

        Modus scripts for ATPG have been written
        for compression mode with fullscan top-off to directory
       'asc/et_scripts_ratio_20'.
        .... Invoke the script from OS command line as 'et -e
        asc/et_scripts_ratio_20/runet.atpg'.

        ----------------------------------------

        Compressing scan chains with ratio 50
        Tool achieved ratio: 41.6

        Modus scripts for ATPG have been written
        for compression mode with fullscan top-off to directory
        'asc/et_scripts_ratio_50'.
        .... Invoke the script from OS command line as 'et -e
        asc/et_scripts_ratio_50/runet.atpg'.
        ----------------------------------------

        Compressing scan chains with ratio 100
        Tool achieved ratio: 62.5

        Modus scripts for ATPG have been written
        for compression mode with fullscan top-off to directory
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
            'asc/et_scripts_ratio_100'.
             .... Invoke the script from OS command line as 'et -e
             asc/et_scripts_ratio_100/runet.atpg'.

     Skipping ATPG because option '-dont_run_atpg' was specified Completed scan compressibility
     analysis for module 'test'
```

- The following command performs an XOR-based compression analysis, with `wide1` masking, with an `broadcast` decompressor, and a ratio list of `"10 20 30"`.

```
legacy_genus:/>  analyze_scan_compressibility -chains 1 \
-library  rules/tsmc13.v -ratio_list "10 20 30" -directory $outdir/newdirnr2
Starting scan compressibility analysis for module 'DLX_CORE'

using existing work directory: './newdirnr2'
Encounter(TM) Test product version information:
Encounter(R) Test and Diagnostics xxxx

Analyzing the design with the following configuration
-----------------------------------------------------
       Design       - DLX_CORE
       Decompressor - broadcast
       Compressor   - xor
       mask         - wide1
       ratio        - 10 20 30

Scan Registers Status
---------------------
       Replacing scan
        Scan flip-flops mapped for DFT          1348        100.00%
       (Number of Scannable Registers) / (Total Register Count) =
       --> 1348 / 1348 = 100%

       (Number of flops mapped to scan) = 1348
       (Number of flops passing tdrc)   = 1348

       (Number of Scannable Registers)
       (Number of flops scan mapped and passing tdrc) +
       (Number of flops in shift-register segments) +
       (Number of flops in abstract scan segments) =
       --> 1348 + 0 + 0 = 1348

       (Total Register Count)
       (Number of registers) +
       (Number of flops in abstract scan segments) =
       --> 1348 + 0 = 1348

       (*) The percentage of scannable registers is 100%.

Building the Fullscan Chains
----------------------------

       Building balanced scan chains assuming all test clocks are compatible and mixing of clock edges in scan chains
       is allowed. If you prefer a different set of constraints, please build the scan chains before calling
       analyze_scan_compressibility.

       Configuring 1 chains for 1348 scan f/f
               ...chain01 of length: 1348

       Modus scripts for ATPG have been written
       for fullscan mode to directory './newdirnr2/et_scripts_fullscan'.

Inserting Scan Compression Logic
--------------------------------

       Compressing scan chains with ratio 10
       Tool achieved ratio: 9.9

       Modus scripts for ATPG have been written
       for compression mode with fullscan top-off to directory
'./newdirnr2/et_scripts_ratio_10'.
       ----------------------------------------

       Compressing scan chains with ratio 20
       Tool achieved ratio: 19.8

       Modus scripts for ATPG have been written
       for compression mode with fullscan top-off to directory './newdirnr2/et_scripts_ratio_20'.
       ----------------------------------------

       Compressing scan chains with ratio 30
       Tool achieved ratio: 29.9
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
       Modus scripts for ATPG have been written
       for compression mode with fullscan top-off to directory './newdirnr2/et_scripts_ratio_30'.

Running Modus ATPG
---------------------------
       Running ./newdirnr2/et_scripts_fullscan/runet.atpg .....
       Running ./newdirnr2/et_scripts_ratio_10/runet.atpg .....
       Running ./newdirnr2/et_scripts_ratio_20/runet.atpg .....
       Running ./newdirnr2/et_scripts_ratio_30/runet.atpg .....

----------------------------------------
Results of analyze_scan_compressibility
----------------------------------------
       Design       - DLX_CORE
       Decompressor - broadcast
       Compressor   - xor
       mask         - wide1

Achieved compression table with fullscan top-off vectors
--------------------------------------------------------
IC    TATR   TDVR    ATCov.   CL       PAT-comp  PAT-fs    Cycles        Runtime    Gatecount    Area
-----------------------------------------------------------------------------------------------------
fs    1.0    1.0     99.99%   1348     -         305       414762        00:00:03   -            -
10    5.6    5.6     99.72%   135      301       22        73532         00:00:04   55           680.656999999992
20    5.5    5.5     99.45%   68       324       37        75711         00:00:05   112          1357.9199999999983
30    4.4    4.4     97.83%   45       299       57        95084         00:00:13   155          1941.826000000001

Achieved compression table without fullscan top-off vectors
-----------------------------------------------------------
IC    TATR   TDVR    ATCov.   CL       PAT-comp  PAT-fs    Cycles
-------------------------------------------------------------------
fs    1.0    1.0     99.99%   1348     -         305       414762
10    9.8    9.8     99.72%   135      301       -         42212
20    17.6   17.6    99.45%   68       324       -         23586
30    27.8   27.8    97.83%   45       299       -         14946


Total ATPG runtime: 00:00:25 (hrs:min:sec).

IC        - Inserted compression
TATR      - Test application time reduction
TDVR      - Test data volume reduction
ATCov.    - ATPG test mode coverage
CL.       - Channel Length
PAT-comp  - Number of compression test patterns
PAT-fs    - Number of fullscan test patterns
Runtime   - ATPG runtime
fs        - Uncompressed (Fullscan) run

Completed scan compressibility analysis for module 'DLX_CORE'
```

- The following command performs MISR-based compression analysis, with `wide1` masking, a ratio list of `2`, sixteen scan chains.

```
legacy_genus:/>  analyze_scan_compressibility -library mylibrary -chains 16 \
-compressor misr -ratio_list 2 -mask wide1 -fault_sample_size 5000 -dir asco
```

Related Information

Analyzing and Reporting Scan Compressibility in *Genus Design for Test Guide for Legacy UI.*

Affected by these commands: connect_scan_chains on page 906
define_dft scan_chain on page 966
Related command: report scan_compressibility on page 697

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

analyze_testability

```
analyze_testability [-library string]
     [-atpg_options string]
     [-build_faultmodel_options string]
     [-build_model_options string]
     [-build_testmode_options string]
     [-fault_sample_size integer]
     [-atpg_log file] -directory string [design]
```

Invokes Modus to perform Automatic Test Pattern Generator (ATPG) based testability analysis in either assume or fullscan mode.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-atpg_options` *`string`*

Specifies a string containing the extra options to run ATPG-based testability analysis.

Note: For more information on these options, refer to the `create_logic_tests` or `create_logic_delay_tests` command in the *Command Line Reference* (of the Modus documentation).

`-atpg_log` *`file`*

Specifies the name of the Modus log file. This file will be generated in the specified directory.

Default: `eta_from_rc.log`

`-build_faultmodel_options` *`string`*

Specifies a string containing the extra options to build a fault model.

Note: For more information on these options, refer to the `build_faultmodel` command in the *Command Line Reference* (of the Modus documentation).

`-build_model_options` *`string`*

Specifies a string containing the extra options to build a model.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: For more information on these options, refer to the `build_model` command in the *Command Line Reference* (of the Modus documentation).

`-build_testmode_options` *`string`*

Specifies a string containing the extra options to build the test modes which define the scan structure and active logic for testing.

Note: For more information on these options, refer to the `build_testmode` command in the *Command Line Reference* (of the Modus documentation).

`design`

Specifies the name of the top-level design on which you want to perform test analysis and test-point selection.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *`string`*

Specifies the working directory for Modus.

`-fault_sample_size` *`integer`*

Specifies the approximate number of faults simulated to predict the test coverage. This number affects the number of partitions or slices to be run depending on the total number of faults in the design.

The default sample size gives a good estimation of fault coverage while limiting the run time. Use a higher number to get a better accuracy, or a smaller number to reduce your run time.

Defaults: `20000`. For full ATPG, `-1`.

`-library` *`string`*

Specifies the list of Verilog structural library files, for example, *`file1 file2`*. Refer to the `write_et_atpg` `-library` option description for additional information.

Note: This option is only required when you invoke this command on a mapped netlist.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Verilog libraries are only needed if the design is mapped to technology gates - otherwise we will use simulation primitives for generic gates, and pass an RTL module description from Genus for the flops and MUXes. Can be used on a mapped design in an assume scan mode. Accuracy improves if analysis is run after you have built the scan chains. You do not need to do a full ATPG analysis: can still do 20k faults. But with the actual chains in place, the results obtained are more accurate than if the analysis was done in an assume scan mode (i.e., without the chains, assuming all the DFFs are in scan chains).

Examples

- The following example performs only ATPG-based testability analysis. The command generates a report on the fault coverage in the log file.

```
analyze_testability
```

- The following command instructs to build a model using the IEEE standard Verilog parser.

```
analyze_testability -build_mode_options "vlogparser=IEEEstandard"
```

- The following command uses the information from Verilog structural library files and writes out ATPG files in the directory specified:

```
analyze_testability -directory atpg_test -library "./LIMS/atpg_library.v"
```

Related Information

Affected by these constraints: define_dft test_mode on page 995
define_dft test_clock on page 990

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_atpg_rules

```
check_atpg_rules [-library string] [-compression]
     [-directory string] [-unix_shell] [design]
```

Generates a template script to run Modus to verify if the design and its test structures are ATPG-ready. More specifically, the generated script allows you to check for

- Tristate drivers for contention

  These conditions might cause manufacturing test problems

- Feedback loops

  Failure to break combinational feedback loops might cause reduced test coverage.

- Clock signal races

  Checks for any flip-flop with a potential race condition between its data and clock signal.

- Test clock control

This command generates the following files:

- `modus.exclude`—A file listing objects to be excluded from the ATPG analysis
- `modus.modedef`—A file describing the test mode when running ATPG in `assumed scan` mode
- *`topmodulename`*`.ASSUMED.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock) and their test function used to build the testmode before actual scan chains exist in the design
- *`topmodulename`*`.FULLSCAN.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock and scan data IOs) and their test function used to build the testmode when actual scan chains exist in the design
- If the ATPG-based testability analysis is run in compression mode, the following pin-assignment files are generated in addition to the *`topmodulename`*`.FULLSCAN.pinassign` file. In this case, All three files include the compression test signals with their appropriate test functions to validate their specific test mode:
  - *`topmodulename`*`.COMPRESSION_DECOMP.pinassign`—A file generated *only* when inserting XOR-based decompression logic
  - *`topmodulename`*`.COMPRESSION.`*`pinassign`*—A file generated to verify the broadcast-based decompression logic

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `runmodus.tsv.tcl`—A template script file for Modus to verify the design and its test structures.

Options and Arguments

`-compression`	Performs additional checks if the design has compression logic.
`design`	Specifies the name of the top-level design for which you want to check if it ATPG-ready. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
`-directory` *`string`*	Specifies the working directory for Modus. **Note:** If the script files already exist, rerunning the command will overwrite them. *Default*: `./check_atpg_rules`
`-library` *`string`*	Specifies the list of Verilog structural library files, for example, *`file1 file2`*. Refer to the `write_et_atpg` `-library` option description for additional information. **Note:** This option is only required when you invoke this command on a mapped netlist.
`-unix_shell`	Writes out the output in shell format.

Example

```
legacy_genus:/> check_atpg_rules

Modus scripts to check whether a design is ATPG ready have been written to directory
'./check_atpg_rules'.
Invoke the scripts as 'modus -e ./check_atpg_rules/runmodus.tsv.tcl'
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_dft_pad_configuration

`check_dft_pad_configuration [design] [> file]`

Checks and reports the data direction control of the pad logic for the test I/O ports.

Options and Arguments

design	Specifies the name of the top-level design to be checked. You should specify this name in case you have multiple top designs loaded. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
file	Specifies the file to which to redirect the detailed output. If no file is specified, the output is written to standard out (`stdout`) and to the log file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_dft_rules

```
check_dft_rules [design] [-advanced]
     [-max_print_violations integer] [> file]
     [-max_print_registers integer]
     [-max_print_fanin integer]
     [-dft_configuration_mode dft_config_mode_name]
```

Evaluates the design for DFT-readiness. Flip-flops that pass the DFT rule checks are later mapped to scan flip-flops during synthesis and included in a scan chain during scan connection. Flip-flops that fail the DFT rule checks and flip-flops marked with either a `dft_dont_scan` attribute or a `preserve` attribute, or flip-flops instantiated in lower-level blocks marked with a `preserve` attribute, are not mapped to scan flip-flops and are excluded from the scan chains. The DFT rule checker also analyzes the libraries and reports on the valid scan cells, and updates the DFT database with all the necessary information about newly added DFT logic and/or DFT signals. Therefore it is considered essential to run the DFT rule checker after every major DFT insertion or definition step.

Table 13-1 lists the DFT rule violations that this command checks..

Table 13-1 Checking For and Auto-Fixing of DFT Rule Violations

DFT Rule Violation	Checked?	Auto-Fixed?
Uncontrollable asynchronous set or reset signals	Checked	Yes
Gated clocks and derived clocks	Checked	Yes
Flip-flop's clock port connected to tied lines	Checked	Yes
Conflicting clock and asynchronous set or reset signals	Checked	No
Tristate contention for internal and external nets	Checked(*)	Yes
Same asynchronous set or reset data race conditions	Checked(*)	Yes
Clock and data race conditions	Checked(*)	No
Floating nets violations	Checked(*)	No
X-source violations	Checked(*)	Yes
Floating conditions	Not checked	No

Note: (*) indicates that the check is performed using the `-advanced` option.

To maximize fault coverage, you should try to fix any DFT rule violations, so that all flip-flops can be included in a scan chain. You can either modify the RTL or use the DFT fix capabilities using the `fix dft violations` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Tip

To include the RTL file name and line number at which the DFT violation occurred in the messages produced by `check_dft_rules`, set the `hdl_track_filename_row_col` root attribute to `true` before elaboration.

You can find the objects created by the `check_dft_rules` command in:

`/designs/`*`design`*`/dft/test_clock_domains`

The detected violations are placed in:

`/designs/`*`design`*`/dft/report/violation`

Options and Arguments

`-advanced`
: Specifies to perform checking for tristate contention, same asynchronous set or reset data race conditions, clock and data race conditions, x-source generators, and floating nets violations.

`design`
: Specifies the name of the top-level design to be checked. You should specify this name in case you have multiple top designs loaded.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_configuration_mode` *`dft_configuration_mode_name`*
: Specifies the object name of the scan mode to be checked. A scan mode is defined using the define_dft dft_configuration_mode command.

  Scan modes can be used to build the top-level scan chains with specific elements in different modes of operation (multi-mode), or when concatenating default scan chains into a single longer scan chain in a different scan mode of operation. After finishing the execution of this command, `report_dft_trace_back` will use this mode by default for all further tracing until another `check_dft_rules` or `set_dft_configuration_mode` is run.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`check_dft_rules` run as a part of a flow is less verbose by default, to save logfile space. Thus, after having run `check_dft_rules`, use `report_dft_violations` when a more verbose report is needed.

file

Specifies the file to which to redirect the detailed output.

`-max_print_fanin` *integer*

Limits the number of pins and ports reported in the fanin cone for any violation.

Note: This option affects DFT checks performed for same asynchronous set or reset data race conditions, clock and data race conditions, and x-source generators.

Default: `20`

`-max_print_registers` *integer*

Limits the number of registers reported for any violation.

Default: `20`

`-max_print_violations` *integer*

Controls the number of DFT violations for which the details are printed to the screen and log file. Specify `-1` to write the details of all violations to the log file.

Default: `20`

Examples

- The following example defines the shift-enable signal and its active polarity, then runs the DFT rule checker. The output of the `check_dft_rules` command is written to the `DFT.rules` file. The return value of the command (2) corresponds to the number of violations found.

```
legacy_genus:/> define_dft shift_enable scan_en -active high
legacy_genus:/> check_dft_rules > DFT.rules
  Checking DFT rules for 'top' module under 'muxed_scan' style

  Checking DFT rules for clock pins
  ...
  Checking DFT rules for async. pins
  ...
Detected 2 DFT rule violation(s)
    ... see the log file for more details
Number of user specified non-Scan registers:   0
      Number of registers that fail DFT rules:   4
      Number of registers that pass DFT rules:   1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
  Percentage of total registers that are scannable: 20%
legacy_genus:/> sh more DFT.rules

Checking DFT rules for 'top' module under 'muxed_scan' style
Processing techlib techlib_25
  Identified a valid scan cell 'SDFFHQX1'
         active clock edge:  rising
  Identified a valid scan cell 'SDFFHQX2'
         active clock edge:  rising
...
Identified 60 valid usable scan cells
Detected 2 DFT rule violation(s)
        Summary of check_dft_rules
        **************************
        Number of usable scan cells: 60
Clock Rule Violations:
----------------------
        Internally driven clock net: 1
            Tied constant clock net: 0
                 Undriven clock net: 0
        Conflicting async/clock net: 0
                    Misc. clock net: 0

Async. set/reset Rule Violations:
---------------------------------
        Internally driven async net: 1
              Tied active async net: 0
                 Undriven async net: 0
                    Misc. async net: 0

   Total number of DFT violations: 2
Clock Violation
# 0: internal or gated clock signal in module 'top', net 'Iclk', inst/pin 'g4/
z' (file: test3.v, line 11) [CLOCK-05]
  Effective fanin cone:
    clk
    en
Async Violation
# 1: async signal driven by a sequential element in module 'top', net 'Iset',
inst/pin 'Iset_reg/q' (file: test3.v, line 13) [ASYNC-05]
  Effective fanin cone:
    Iset_reg/q
        Violation # 0 affects  4 registers
        Violation # 1 affects  4 registers

        Note - a register may be violating multiple DFT rules
Total number of Test Clock Domains: 1
  DFT Test Clock Domain: clk
        Test Clock 'clk' (Positive edge) has   1 registers
Number of user specified non-Scan registers:   0
    Number of registers that fail DFT rules:   4
    Number of registers that pass DFT rules:   1
Percentage of total registers that are scannable: 20%
```

- The following example shows tristate net checking with the `check_dft_rules -advanced` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> check_dft_rules -advanced
Checking DFT rules for 'test' module under 'muxed_scan' style
    Processing techlib tsmc_25 for muxed_scan scan cells
    Identified 60 valid usable scan cells
  Checking DFT rules for clock pins
Info    : Added DFT object. [DFT-100]
        : Added test clock domain 'clk'.
Info    : Added DFT object. [DFT-100]
        : Added test clock 'clk'.
  Checking DFT rules for async. pins
  Checking DFT rules for shift registers.
  Checking DFT rules for tristate nets.
  Checking DFT rules for clock data race conditions.
  Checking DFT rules for set reset data race conditions.
  Checking DFT rules for x-sources.
Detected 1 DFT rule violation(s)
        Summary of check_dft_rules
        **************************
        Number of usable scan cells: 60
Clock Rule Violations:
----------------------
          Internally driven clock net: 0
              Tied constant clock net: 0
                   Undriven clock net: 0
        Conflicting async & clock net: 0
                      Misc. clock net: 0

Async. set/reset Rule Violations:
---------------------------------
        Internally driven async net: 0
              Tied active async net: 0
                 Undriven async net: 0
                    Misc. async net: 0

Advanced DFT Rule Violations:
-----------------------------------
         Tristate net contention violation: 1
        Potential race condition violation: 0
                       X-source violation: 0

   Total number of DFT violations: 1


Warning : DFT Tristate net contention Violation. [DFT-315]
        : # 0 <vid_0_tristate_net>: tristate net 'tbus' connected to pin
 'out_reg/d' potentially driven by conflicting values   [TRISTATE_NET-01]
 : To remove the net contention violation in scan-shift mode, either modify the
RTL, or use the '-tristate_net' option of the 'fix_dft_violations' command.

    Tristate net drivers:
      i_block2/g1/z
      i_block1/g1/z
      i_block3/g1/z

    Violations sorted by type and number of affected registers
    Note - a register may be violating multiple DFT rules.

        There are '1' tristate net violations.

----------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*:

- Running the DFT Rule Checker
- Advanced DFT Rule Checking
- Defining Scan Configuration Modes
- Concatenating Scan Chains

Affected by these constraints:	define_dft abstract_segment on page 915
	define_dft dft_configuration_mode on page 925
	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995
Affects these commands:	connect_scan_chains on page 906
	fix_dft_violations on page 1010
	report dft_registers on page 1128
	syn_map on page 529
	syn_opt on page 531
Affected by these attributes:	dft_controllable
	dft_dont_scan
	dft_identify_test_signals
	dft_identify_top_level_test_clocks
	dft_scan_style
	(instance) preserve
	(subdesign) preserve
	dft_identify_xsource_violations_from_timing_models
Sets these attributes:	dft_status
	dft_violation
	type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Violations Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_dft_setup

```
check_dft_setup
     [-2d_compression] [design]
```

Checks the DFT setup of the specified features for correctness and compatibility.

Options and Arguments

`-2d_compression`	Checks the DFT setup of 2D compression.
`design`	Specifies the design for which to check the DFT setup.

Related Information

Affects this command: syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

compress_block_level_chains

```
compress_block_level_chains
     {-ratio integer | -channel_length integer}
     [-chains actual_scan_chain]...
     [-decompressor {broadcast | xor }]
     [-compressor xor [-mask {wide1|wide2}]
     |-compressor misr [-mask {wide0 | wide1 | wide2}]
     |-compressor hybrid [-mask {wide0 | wide1 | wide2}] ]
     [-mask_sharing_ratio integer]
     [-power_aware] [-target_period integer]
     [-preview] [-inside instance] [design]
```

Adds decompression and compression logic to reduce the effective length of the actual scan chains in the specified block. Use this command instead of the `compress_scan_chains` command when the block to be compressed will be modeled as a compressed block in the hierarchical compression flow.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-chains` *`actual_scan_chain...`*

Specifies the names of the actual scan chains to be compressed.

By default all actual scan chains are compressed.

Note: When inserting `MISR` logic, all scan chains must eventually be compressed.

`{-channel_length` *`integer`* `| -ratio` *`integer`*`}`

Controls the maximum length of the internal scan channels. You can either specify the maximum length directly or the tool can derive the length of the internal scan channels by dividing the longest actual scan chain length by the ratio.

Note: The `-channel_length` option is recommended for the hierarchical compression flow.

`-compressor {xor | misr}`

Specifies the type of compression logic to be built:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `xor` specifies to build an XOR-based compressor
- `misr` specifies to build a `MISR-based` compressor
- `hybrid` specifies to build a `MISR` compression with `MISR` bypass capability. Bypassing the MISR allows you to perform compression using just the XOR compressor.

Default: `xor`

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic to be built:

- `xor` specifies to build an XOR-based spreader network in addition to the broadcast-based decompression logic
- `broadcast` specifies to build a broadcast-based decompression logic (simple scan fanout).

Default: `broadcast`

`design`

Specifies the name of the top-level design whose scan chains must be compressed.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-inside` *`instance`*

Specifies the instance in which to instantiate the compression logic.

By default, the compression logic is inserted as a hierarchical instance in the top-level of the design.

`-mask {wide0| wide1 | wide2}`

Inserts scan channel masking logic of the specified type.

The masking types that can be used depend on the compressor type specified with the `-compressor` option.

By default, no masking logic is inserted.

Note: The syntax indicates which types are available for each of the compressor types.

`-mask_sharing_ratio` *`integer`*

Specifies the number of internal scan channels sharing a mask register. The specified integer may not exceed the value specified for the compression ratio.

Note: This option is only valid with `wide1` and `wide2` masking.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-power_aware`
: Ensures that the mux logic added during compression obeys the boundaries of the power domains defined in the CPF file.

`-preview`
: Reports the requested ratio, the maximum original scan chain length, the maximum subchain length, and the number of internal scan channels that would be created without making modifications to the netlist. Use this option to verify your compression architecture prior to inserting the compression logic.

`-target_period` *`integer`*
: Specifies a target clock period (in picoseconds) used to optimize the compression macro. If the value zero (`0`) is specified, synthesis is performed with a low effort compile, and without applying external (input/output delay) constraints.

  Default: value for `test_clock` period of the scan chains being compressed

Example

The following command requests an XOR-based compression with masking logic of type 1 and decompression logic of type XOR for design core.

```
compress_block_level_chains -channel_length 7 -decompressor xor -mask wide1 core
```

Related Information

Affected by these constraints:	define_dft jtag_instruction on page 933
	define_dft shift_enable on page 978
	define_dft test_mode on page 995
Affected by these commands:	connect_scan_chains on page 906
Affects these commands:	report dft_chains on page 1125
	write_dft_abstract_model on page 1162
Sets these attributes:	compressed
	dft_compression_signal
	dft_mask_clock

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dft_misr_clock

type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

compress_scan_chains

```
compress_scan_chains {-ratio integer | -channel_length integer]
     [-chains actual_scan_chain]...
     [-auto_create] [-power_aware]
     [-decompressor {broadcast
          | xor [-spread_enable test_signal]}]
     [-compression_enable test_signal] [-target_period integer]
     [-master_control test_signal]
     [ -compressor xor
          -mask {wide1|wide2} [-mask_clock {port|pin} [-allow_shared_clock]]
          [-mask_load test_signal] [-mask_enable test_signal]
          [-create_mask_or_misr_chain
               -mask_or_misr_sdi {pin|port}
               -mask_or_misr_sdo {pin|port} [-shared_output]]
          [-mask_sharing_ratio integer]
          [-apply_timing_constraints [-timing_mode_names mode_list]]
          [-write_timing_constraints file]
          [-low_pin_compression
            [-low_pin_comp_control shift_enable] [-shift_enable shift_enable]]
     | -compressor misr
          [-serial_misr_read [-misr_observe test_signal]]
          [-misr_clock {port|pin}]
          {-misr_reset_enable test_signal | -misr_reset_clock test_signal}
          [-misr_read test_signal | -use_all_scan_ios_unidirectionally]
          [-misr_shift_enable test_signal]
          [-mask {wide0 | wide1 | wide2}
          [-mask_sharing_ratio integer
          [-mask_clock {port|pin} [-allow_shared_clock]][-mask_load test_signal]
          [-mask_enable test_signal_list]
          [-create_mask_or_misr_chain
               -mask_or_misr_sdi {pin|port}
               -mask_or_misr_sdo {pin|port} [-shared_output]]
     | -compressor hybrid
          [-misr_bypass test_signal]
          [-serial_misr_read] [-misr_observe test_signal]
          [-misr_clock {port|pin}]
          {-misr_reset_enable test_signal | -misr_reset_clock test_signal}
          [-misr_shift_enable test_signal]
          [-mask {wide0 | wide1 | wide2}]
          [-mask_sharing_ratio integer]
          [-mask_clock {port|pin}] [-mask_load test_signal]
          [-mask_enable test_signal_list]
          [-create_mask_or_misr_chain
               -mask_or_misr_sdi {pin|port}
               -mask_or_misr_sdo {pin|port} [-shared_output]]]
     | -compressor smartscan_xor
          -smartscan_ratio integer
          [-gate_shared_compression_clock] [-smartscan_no_update_stage]
          [-smartscan_serial_only | -smartscan_serial_scan_ins {pins|ports}]
          [-smartscan_parallel_access test_signal]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
        [-smartscan_enable test_signal]
        [-smartscan_pulse_width_multiplier {1|2|4}]
        [-mask {wide1|wide2}] [-mask_clock {port|pin}]
        [-mask_load test_signal]
        [-mask_sharing_ratio integer]
        [-apply_timing_constraints [-timing_mode_names mode_list]]
        [-write_timing_constraints file] ]
    [-preview] [-inside instance] [design]
    [-jtag_control_instruction jtag_instruction]
        [-allow_multiple_jtag_control]]
    [-share_mask_enable_with_scan_in]
```

Adds decompression and compression logic to reduce the effective length of the actual scan chains. This is usually invoked in a top-down design flow that does not use hierarchical compression and requires flat ATPG.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-allow_multiple_jtag_control`

When controlling the compression testmode from a JTAG macro, (that is, the `-jtag_control_instruction` option is specified), `compress_scan_chains` checks for the presence of other compression macros that are also JTAG controlled. RC-DFT does not automatically support such a configuration and therefore insertion of a second JTAG controlled compression macro is disallowed by default. Specify this option to bypass this check and insert additional JTAG controlled compression macros. When this option is specified, the tool assumes you will manually perform any additional stitching needed and will appropriately modify any files generated for Modus.

Note: You must also specify the `-jtag_control_instruction` option with this option.

`-allow_shared_clock`

Allows the mask or `MISR` clock to be shared with an existing full-scan test clock.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: The mask or `MISR` clock can only be shared with a test clock if you added gating logic which prevents the scan flops from pulsing during the channel mask load or `MISR` reset sequences. Since functional clocks are typically used for scanning, this requirement means that the functional clocks must be gated during test.

Important

When specified with the `-auto_create` option, a `-mask_load` pin is automatically created. The mask_load pin must additionally be used to gate off the clock being shared. If this gating logic is not added, the mask loading or MISR reset procedure will corrupt the test data in the design.

`-apply_timing_constraints`

Applies timing constraints to the appropriate compression control signals to prevent the mapper from considering these paths for timing optimization.

Timing constraints will be applied in all user-specified timing modes.

Note: If your design has multiple timing modes but you did not specify the `-timing_mode_names` option to list the timing modes for which to write the constraints, no additional constraints are applied.

`-auto_create`

Automatically creates the necessary test pins as top-level ports.

If you omitted any of the following options `-compression_enable`, `-spread_enable`, `-mask_clock`, `-mask_load`, `-mask_enable`, `-mask_or_misr_sdi`, `-mask_or_misr_sdo`, `-misr_clock`, `-misr_observe`, `-misr_reset_enable`, `-misr_read`, `-misr_bypass`, `-smartscan_enable`, and `-smartscan_parallel_access`, the appropriate ports will be created and named using the following format:

`prefixOption`

For example, *`prefix`*`compression_enable`, where *`prefix`* is the value of the `dft_prefix` root attribute.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-chains` *`actual_scan_chain...`*

Specifies the names of the actual scan chains to be compressed.

By default all actual scan chains are compressed.

Note: When inserting `MISR` logic, all scan chains must eventually be compressed.

`{-channel_length` *`integer`* `| -ratio` *`integer`*`}`

Controls the maximum length of the internal scan channels. You can either specify the maximum length directly or the tool can derive the length of the internal scan channels by dividing the longest actual scan chain length by the ratio.

Note: The `-channel_length` option is recommended for the hierarchical compression flow.

`-compression_enable` *`test_signal`*

Specifies the name of the test signal that enables configuring the actual scan chains in compression mode.

Note: If you do not specify the `-auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. If you request to build the compression logic with a master control signal (`-master_control`), the input port driving the compression enable signal can be an existing functional pin (specified through the `-shared_in` option of `define_dft test_mode`). If you do not specify the `-master_control` option, you must define the compression enable signal without the `-shared_in` option.

`-compressor {xor | misr| hybrid | smartscan_xor}`

Specifies the type of compression logic to be built:

- `xor` specifies to build an XOR-based compressor
- `misr` specifies to build a `MISR-based` compressor
- `hybrid` specifies to build a `MISR` compression with `MISR` bypass capability. Bypassing the MISR allows you to perform compression using just the XOR compressor.
- `smartscan_xor` specifies to include smartscan logic in the compression macro.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: `xor`

`-create_mask_or_misr_chain`

Specifies to build a separate full-scan chain for the mask and `MISR` registers.

Note: To place the mask or `MISR` registers in a separate chain, an additional scan data input pin and scan data output pin are needed. You can either specify these pins using the `-mask_or_misr_sdi` and `-mask_or_misr_sdo` options, or make sure that the `-auto_create` option is specified.

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic to be built:

- `xor` specifies to build an XOR-based spreader network in addition to the broadcast-based decompression logic
- `broadcast` specifies to build a broadcast-based decompression logic (simple scan fanout).

Default: `broadcast`

`design`

Specifies the name of the top-level design whose scan chains must be compressed. You should specify this name in case you have multiple top designs loaded.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-gate_shared_compression_clock`

Specifies that the clock to the compression macro is shared with the test clock to the scan chains and that the tool must insert gating logic on the test clock path. The test clock will be gated using the `TEST_CLOCK_ENABLE` pin on the compression macro.

Note: The `TEST_CLOCK_ENABLE` pin will always exist on the smartscan compression macro and can be used for the clock gating. If this gating logic is not added, the scan chains will be disturbed during the mask load and/or smartscan register load operations.

Note: If you omit this option, you must manually insert the gating logic. The tool will print a message to warn you that this connection must be made but will not perform any subsequent checks to ensure the gating logic has been correctly added.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-inside` *`instance`*

Specifies the instance in which to instantiate the compression logic.

By default, the compression logic is inserted as a hierarchical instance in the top-level of the design.

`-jtag_control_instruction` *`jtag_instruction`*

Specifies which JTAG instruction will be used to target the compression macro's test data register.

`-low_pin_comp_control` *`shift_enable_signal`*

Specifies a shift-enable signal used to control low pin count compression.

You cannot specify the same shift-enable signal for both the `-low_pin_comp_control` and `-shift_enable` options.

If you omit this signal, the tool will use one of the default shift-enable signals.

If no shift-enable pin exists, the tool creates an LPC_CONTROL shift-enable pin if the `-auto_create` option is specified.

`-low_pin_compression`

Reduces the number of compression control pins required by using encoded control signals.

`-mask {wide0| wide1 | wide2}`

Inserts scan channel masking logic of the specified type.

The masking types that can be used depend on the compressor type specified with the `-compressor` option.

By default, no masking logic is inserted.

Note: The syntax indicates which types are available for each of the compressor types.

`-mask_clock {`*`pin`*`|`*`port`*`}`

Specifies the clock that controls the mask registers.

Note: The input port associated with this option can be an existing functional pin. This clock cannot be shared with an existing full-scan test clock pin unless you also specify the `-allow_shared_clocks` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-mask_enable` *`test_signal`*

Specifies the name of the test signal that controls whether mask bits should be applied during the current scan cycle.

For `wide2` masking, two mask enable signals must be specified.

This option cannot be specified when the `-compressor` option is set to `smartscan_xor`.

Note: If you do not specify the `-auto_create` option, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-mask_load` *`test_signal`*

Specifies the name of the test signal that enables loading of the mask data into the mask data registers.

Note: If the mask_clock is dedicated (that is, is only used for mask register loading), this signal is not needed. If this signal is shared (is used to clock the MISR or other logic in the circuit), this signal is needed to gate non mask load clock pulses from corrupting the mask registers. If the `mask_clock` is shared with other logic, you can use this signal to protect the shared logic from corruption during the mask load sequence.

`-mask_or_misr_sdi {`*`pin`*`|`*`port`*`}`

Specifies the scan data input pin or port of the mask or `MISR` chain.

Note: The input port associated with this option can be an existing functional pin.

`-mask_or_misr_sdo {`*`pin`*`|`*`port`*`}`

Specifies the scan data output pin or port of the mask or `MISR` chain.

Note: If the output port associated with this option is an existing functional pin, you must specify the `-shared_out` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-mask_sharing_ratio` *`integer`*

Specifies the number of internal scan channels sharing a mask register. The specified integer may not exceed the value specified for the compression ratio.

Note: This option is only valid with `wide1` and `wide2` masking.

`-master_control` *`test_signal`*

Specifies the master control signal that gates the compression enable signal used for compression.

Note: This test signal must be dedicated for test and must have been defined using the `define_dft test_mode` command.

`-misr_bypass` *`test_signal`*

Specifies the test signal used to bypass the `MISR-based` logic. This test signal is required in hybrid compression mode.

Note: If you do not specify the -`auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-misr_clock {`*`pin`*`|`*`port`*`}`

Specifies the clock that controls the `MISR` registers.

Note: This input port cannot be shared with an existing full-scan test clock unless it is only used to accumulate the MISR signature or if the `-allow_shared_clocks` option is specified. The `-misr_clock` option is only used to accumulate the MISR signature if there are separate `-mask_clock` and `-misr_reset_clock` signals specified.

`-misr_observe` *`test_signal`*

Specifies the test signal used to select Serial MISR Read. This is required when the `-serial_misr_read` option is specified unless `-auto_create` or `-jtag_control_instruction` is also specified.

Note: You must also specify the `-serial_misr_read` option with this option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-misr_read` *`test_signal`*

Specifies the test signal to configure any bidirectional scan I/O pads for MISR compression.

Note: This option is mutually exclusive with the `-use_all_scan_ios_unidirectionally` option. Using scan I/O bidirectionally during MISR compression is only available with the -compressor misr option. When using the `-compressor hybrid` option, all scan I/O are used unidirectionally.

`-misr_reset_clock` *`test_signal`*

Specifies a separate dedicated test signal that is used to asynchronously reset the MISR.

Note: This option is mutually exclusive with the `-misr_reset_enable` option.

`-misr_reset_enable` *`test_signal`*

Specifies the test signal used to reset the `MISR` registers.

Notes:

- This option is mutually exclusive with the `-misr_reset_clock` option.
- If you do not specify the -`auto_create` option, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-misr_shift_enable` *`test_signal`*

Specifies the test signal used to enable MISR accumulation during scan shifting.When this signal is de-asserted, the contents of the MISR register will not change.

Note: If this option is omitted, the default shift-enable test signal for the design is used. If this option is specified, you must have defined this test signal using the `define_dft shift_enable` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft shift_enable` command).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-power_aware`
: Ensures that the mux logic added during compression obeys the boundaries of the power domains defined in the CPF file.

`-preview`
: Reports the requested ratio, the maximum original scan chain length, the maximum subchain length, and the number of internal scan channels that would be created without making modifications to the netlist. Use this option to verify your compression architecture prior to inserting the compression logic.

`-serial_misr_read`
: Specifies to include support for reading MISR bits serially through the scan data pins.

`-share_mask_enable_with_scan_in`
: Allows to share the mask enable port with a scan data in port. When you specify this option you enable insertion of an asymmetrical compression macro.

`-shared_output`
: Specifies that the scan data output port of the dedicated mask or `MISR` chain must be shared with a functional port.

  The tool creates the additional logic required to share the port.

`-shift_enable` *`shift_enable`*
: Specifies the shift-enable signal to be used for low pin count compression.

  If you omit this option, the tool will use the default shift-enable signal.

`-smartscan_enable` *`test_mode_signal`*
: Specifies the test mode signal that enables the smartscan mode. You can omit this option if you specified the `-auto_create` option. The tool will create the `smartscan_enable` port on the compression macro and connect it to the primary input for this test mode signal.

  When this signal is active it ensures that the smartscan mode is ON. Currently when this signal is inactive, the smartscan flops are part of the fullscan chains only. When inactive, and in the compression modes, the smartscan flops are *not* part of the compression channels.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-smartscan_no_update_stage`

Prevents the insertion of update registers between the deserializer and the decompressor. In this case, lockup latches are inserted between the deserializer flops and the decompressor.

You cannot specify this option when you have set the `-smartscan_pulse_width_multiplier` option to either `2` or `4`.

By default, the tool inserts update registers between the deserializer and the decompressor.

`-smartscan_parallel_access` *`test_mode_signal`*

Specifies the test mode signal to use for parallel access to the smartscan flops. You can omit this option if you specified the `-auto_create` option. The tool will create the `smartscan_parallel_access` port on the compression macro and connect it to the primary input for this test mode signal.

`-smartscan_pulse_width_multiplier {1|2|4}`

Determines whether to add clock divider logic to widen the clock pulse going to the scan chains. You can specify the following values:

- 1—no logic added
- 2—increases the scan path through the SmartScan clock controller with 1 bit
- 4—increases the scan path through the SmartScan clock controller with 2 bits

Default: `1`

`-smartscan_ratio` *`integer`*

Specifies the number of parallel scan data input pins that correspond to a single serial scan data input pin. The number of defined (fullscan) chains must be an integral multiple of the specified smartscan ratio.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-smartscan_serial_only`

Specifies to only insert the smartscan serial-only interface. The number of deserializer (and serializer) registers will match the number of the defined chains. When building the model for Modus (during the `build_model` step in the `write_et_atpg` scripts), the tool will create the pseudo pins for the parallel interface.

`-smartscan_serial_scan_ins {`*`pins`*`| `*`ports`*`}`

Specifies the pins or ports to be used as serial scan data inputs for the SmartScan serial and parallel flow.

The scan data outputs of the scan chains starting at these scan data inputs will be used as serial scan data outputs.

The number of pins or ports specified must match the number of serial scan data inputs required for the SmartScan architecture as determined by the number of fullscan chains and the `-smartscan_ratio`. You cannot specify a pin or port that is not a scan data input of any of the scan chains that are part of the compression. Make sure that all scan data input ports for OPCG-side scan chains are specified with this option.

Note: This option cannot be specified together with the `-smartscan_serial_only` option.

`-spread_enable` *`test_signal`*

Specifies the name of the test signal that enables applying the input test data to an XOR-based spreader network.

Use this option when `-decompressor` is set to `xor`.

Note: If you do not specify the -`auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-target_period` *`integer`*

Specifies a target clock period (in picoseconds) used to optimize the compression macro. If the value zero (`0`) is specified, synthesis is performed with a low effort compile, and without applying external (input/output delay) constraints.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: value for `test_clock` period of the scan chains being compressed

`-timing_mode_names` *`mode_list`*

Specifies the timing modes for which to generate the additional timing constraints that apply to the compression control signals.

The timing modes are taken into account when you specify the `-apply_timing_constraints` and `-write_timing_constraints` options.

Note: This applies only to multi-mode designs. Modes are created with the `create_mode` command.

`-use_all_scan_ios_unidirectionally`

Disables use of bidirectional scan I/O for a `MISR-based` compressor.

Note: This option is mutually exclusive with the `-misr_read` option.

`-write_timing_constraints` *`file`*

Specifies the file to which to write the timing constraints applied to the appropriate compression control signals to prevent the mapper from considering these paths for timing optimization. The timing constraints are not written if you specify the `-preview` option.

Timing constraints will be applied in all user-specified timing modes.

Note: If your design has multiple timing modes but you did not specify the `-timing_mode_names` option to list the timing modes for which to write the constraints, constraints for all modes are written to the file.

Examples

- The following command requests XOR-based compression logic without masking, and requests creation of the necessary ports for the required test signals. In this case only a compression enable signal is required and thus one test port is created.

```
legacy_genus:/> compress_scan_chains -ratio 5 -compressor xor -auto_create
Will create a test port for 'compression_enable'
Checking out license 'Encounter_Test_Architect'... (1 seconds elapsed)
...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command requests XOR-based compression logic with masking logic of type `wide1`, and requests creation of the necessary ports for the required test signals. In this case three extra test signals are required for the masking logic and thus four test ports are created.

```
legacy_genus:/> compress_scan_chains -ratio 5 -compressor xor -mask wide1 \
 -auto_create
Will create a test port for 'compression_enable'
Will create a test port for 'mask_load'
Will create a test port for 'mask_enable'
Will create a test port for 'mask_clock'
Checking out license 'Encounter_Test_Architect'... (1 seconds elapsed)
....
```

- The following command requests an XOR-based compression with masking logic of type `wide1`, with decompression logic of type `xor`, and requests creation of the necessary ports for the required test signals. Compared to the second example, one extra test signal— the `spread_enable` signal—is required for the decompression logic, and thus five test ports are created.

```
legacy_genus:/> compress -ratio 5 -compressor xor -decompressor xor \
-mask wide1 -auto_create
Will create a test port for 'compression_enable'
Will create a test port for 'spread_enable'
Will create a test port for 'mask_load'
Will create a test port for 'mask_enable'
Will create a test port for 'mask_clock'
Checking out license 'Encounter_Test_Architect'... (2 seconds elapsed)...
```

- The following command requests a `MISR`-based compression with masking logic of type `wide1`, and requests creation of the necessary ports for the required test signals. Compared the two second example, one extra test signal—the `misr_reset_enable` signal—is required to reset the `MISR` registers, and thus five test ports are created.

```
legacy_genus:/> compress -ratio 5 -compressor misr -mask wide1 -auto_create
Will create a test port for 'compression_enable'
Will create a test port for 'misr_reset_enable'.
Will create a test port for 'misr_clock'
Info - Defaulting the misr shift enable signal to the default shift_enable /
designs/test/dft/test_signals/SE
Will create a test port for 'mask_load'
Will create a test port for 'mask_enable'
Info - will share the 'misr_clock' and the 'mask_clock' ...
```

Related Information

Affected by these constraints: define_dft jtag_instruction on page 933
define_dft shift_enable on page 978
define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Affected by these commands:	connect_scan_chains on page 906
Affects this command:	report dft_chains on page 1125
Sets these attributes:	compressed
	dft_compression_signal
	dft_mask_clock
	dft_misr_clock
	type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

concat_scan_chains

```
concat_scan_chains
     -name string
     -chains actual_scan_chains
     [-dft_configuration_mode dft_config_mode_name]
     [-preview] [design]
```

Inserts muxing logic into the scan path of actual scan chains such that the scan chains are concatenated to become a single, longer scan chain in a specific test mode of operation.

Note: An actual scan chain may be specified only once per test mode, that is, multiple configurations of the same scan chain in the same test mode are disallowed.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-chains` *`scan_chains`*

Specifies the names of the actual scan chains to be concatenated; where the scan chains are concatenated in the specified order.

`design`

Specifies the name of the top-level design for which to concatenate the scan chains. Specify this name if you have multiple top designs loaded.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_configuration_mode` *`dft_configuration_mode_name`*

Specifies the object name of the scan mode used to concatenate the actual scan chains.

`-name` *`string`*

Specifies the name of the concatenated chain.

`-preview`

Reports how the actual scan chains would be concatenated, but does not perform the concatenation.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these constraints:	define_dft dft_configuration_mode on page 925
	define_dft shift_enable on page 978
	define_dft test_mode on page 995
	set_compatible_test_clocks on page 1140
Affected by this command:	check_dft_rules on page 870
Affects these commands:	report dft_chains on page 1125
	write_dft_abstract_model on page 1162
	write_scandef on page 1216
Sets these attributes:	Actual Scan Chain attributes
	Actual Scan Segment attributes
Affected by these attributes:	decoded_pin
	dft_lockup_element_type
	dft_mix_clock_edges_in_scan_chains

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

configure_pad_dft

```
configure_pad_dft -mode {input | output | tristate}
     -test_control test_signal port
```

Inserts the required logic to configure the data direction control for a bidirectional or tristate pad during test mode. This command can be used after running the `check_dft_pad_configuration` command to fix the pads configuration.

Note: This command can configure a generic pad.

Options and Arguments

`-mode {input | output | tristate}`

Specifies in which mode the pad must be configured in test mode.

`input`	Specifies to configure the pad in input mode.
`output`	Specifies to configure the pad in output mode.
`tristate`	Specifies to disable the pad.

`port`

Specifies the top-level port that is connected to the I/O pad that the Genus-DFT engine needs to configure.

`-test_control` *`test_signal`*

Specifies the test signal to use to control the pad.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

Related Information

Affected by these constraints:	define_dft shift_enable on page 978
	define_dft test_mode on page 995
Affects these commands:	check_dft_rules on page 870
	connect_scan_chains on page 906

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_cgic_test_pins_controlling_wrapper_cells

```
connect_cgic_test_pins_controlling_wrapper_cells
     -test_enable {pin|port}
     [-genus_cgics_only] [-verbose]
     [design | instance]
```

Identifies the integrated clock-gating cells which control the clock pin of shared wrapper cells inserted by the `insert_dft wrapper_cell` command. The test pin of each clock-gating cell will be rewired to use the control signal specified with the `-test_enable` option.

The test signal should have the function of `shift_enable | EXTEST`. By doing so, you ensure that the clock gates can propagate the clocks to the shared wrapper cells in an EXTEST mode. This command should be run after the wrapper cells have been added to the design.

Options and Arguments

design|*instance*	Specifies the design or instance in which you want to identify the integrated clock-gating cells.
`-genus_cgics_only`	Limits the identification to Genus-inserted integrated clock-gating cells. If you omit this option, user-instantiated (or architectural) clock-gating cells in the fanin of the Genus-inserted clock-gating cells will be included in the list of cells whose test pins will be rewired.
`-test_enable {`*pin*|*port*`}`	Connects the test pin of the integrated clock-gating cells controlling the shared wrapper cells to the specified pin or port
`-verbose`	Prints a table showing the leaf clock gates and the shared wrapper flops controlled by the clock gate.

Related Information

Related command: insert_dft wrapper_cell on page 1094

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_compression_clocks

```
connect_compression_clocks
     [-mask_clock test_clock] [-misr_clock test_clock]
     [design]
```

Connects the compression clocks that were auto-created using the `compress_block_level_chains` command at the block level to the compression clocks at the top level.

This command applies only to the hierarchical compression flow where the top level compression is added using the compress_scan_chains command.

Note: This command does not apply to the core or top level processing flows documented in the Hierarchical Test.

Options and Arguments

design
: Specifies the name of the design for which to connect the compression clocks.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

  This option is required if multiple designs are loaded.

-mask_clock *test_clock*
: Specifies the top-level clock that controls the mask registers.

  If you omit this option, the tool uses the top-level test clock whose `dft_mask_clock test_clock` attribute is set to `true`.

-misr_clock *test_clock*
: Specifies the top-level clock that controls the `MISR` registers.

  If you omit this option, the tool uses the top-level test clock whose `dft_misr_clock test_clock` attribute is set to `true`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands: compress_block_level_chains on page 878
compress_scan_chains on page 882

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_dft_top_level

```
connect_dft_top_level
     [design][-scan_in_pipeline_depth integer]
     [-scan_out_pipeline_depth integer]
```

Connects the top-level test signals to the top level Test Interconnect Macro and the test signals defined for the scan abstracted cores in the Hierarchical Test flow. This command is run after inserting the test compression logic and is one of the final processing steps before writing the Modus database in the Hierarchical Test Flow: Integrating the Top level.

Options and Arguments

design
: Specifies the name of the design for which to connect the top-level test interconnect macro.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

  This option is required if multiple designs are loaded.

-scan_in_pipeline_depth *integer*
: Specifies the length of the input pipeline.

-scan_out_pipeline_depth *integer*
: Specifies the length of the output pipeline.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_opcg_segments

```
connect_opcg_segments
     [-chains actual_scan_chains] [-side_scan_only]
     [-use_ports_for_side_scan_connections port_list]
     [-preview] [design]
```

Connects the scan segments associated with the `OPCG` logic (OPCG programming registers and OPCG domain macros) into the actual scan chains so that OPCG side chains are created in order to load/program the OPCG. You must run this command after you have connected the scan chains.

There is no need to run this command if compression has been inserted using `insert_test_compression` when a `test_signal` with function `opcg_load` was present.

Options and Arguments

design
: Specifies the name of the design for which to connect the `OPCG` segments.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

  This option is required if multiple designs are loaded.

`-chains` *actual_scan_chains*
: Specifies the names of the actual scan chains to which the `OPCG` segments must be prepended.

`-preview`
: Shows the potential connections for the `OPCG` segments, without making any modifications to the netlist.

`-side_scan_only`
: Connects just the OPCG side-scan while the OPCG scan segments are included in the scan chains by executing `connect_scan_chains -include_opcg_segments`.

`-use_ports_for_side_scan_connections` *port_list*
: Specifies the ports to be used as scan data input pins when making the connections for the side scan chains.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Use this option for RTL inserted scan compression flows in which the actual chains are built with respect to scan compression channels and not as fullscan chains. In this flow, the specified ports will be used to connect the side scan chains instead of using the internal scan-data input pins specified to build the scan compression channels.

Note: You can also use this option in a flow in which you build your fullscan chains followed by scan compression. The specified ports will be used to connect the side scan chains instead of using the scan-data input pins defined for the fullscan chains.

Related Information

Affected by these commands	connect_scan_chains on page 906
	insert_dft opcg on page 1063

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_scan_chains

```
connect_scan_chains [design]
     [-preview] [-auto_create_chains]
     [-incremental] [-chains chain_list]
     [-elements element_list]
     [-keep_connected_SE]
     [-dont_exceed_min_number_of_scan_chains]
     [-pack | -create_empty_chains]
     [-dft_configuration_mode dft_config_mode_name]
     [-physical [-cluster_aggressively_high]]
     [-update_placement]
     [-include_opcg_segments]
     [-power_domain power_domain_list] [-zipper_stitch]]
```

Configures and connects scan flip-flops which pass the DFT rule checks into scan chains. This command works at the current level of the hierarchy and all lower hierarchies instantiated in this module. The design must be mapped to the target library before connecting scan chains in a design.

The command returns the number of scan chains that the scan configuration engine creates (or would create if you use the `-preview` option).

You can find the objects created by the `connect_scan_chains` command in:

```
/designs/design/dft/actual_scan_chains
/designs/design/dft/actual_scan_segments
```

Options and Arguments

`-auto_create_chains`	Allows the scan configuration engine to add new chains that are not defined through a `define_dft scan_chain` constraint. Without this option, the scan configuration engine reports an error if it needs more scan chains than have been defined with the `define_dft scan_chain` command.
`-chains` *`chain_list`*	Connects only the specified user-defined chain names. If the list is empty, none of the user-defined chains can be connected at this time. New chains are created if you specify the `-auto_create_chains` option. The specified user-defined chains must have been defined using a `define_dft scan_chain` constraint. If omitted, all user-defined chains can be connected.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-cluster_aggressively_high`

Uses the physical location of the scan flops to build scan chains for improved scan chain balancing and scan wirelength reduction. This option must be specified with the `-physical` option.

`-create_empty_chains`

Allows the scan configuration engine to create empty scan chains by making a direct connection from their scan data input to their scan data output if the number of scan chains to be configured is less than the minimum number of scan chains required in the design.

Note: Do not use this option when configuring scan chains to be used as internal scan channels that are loaded and unloaded using on-chip compression logic.

If this option is not specified, the scan configuration engine will move scan flops between compatible chains to satisfy the minimum number of scan chains requirement. This can result in configured scan chains having a sequential depth of one element.

`-dft_configuration_mode` *`dft_configuration_mode_name`*

Specifies the scan mode for which to build the scan chains.

`-dont_exceed_min_number_of_scan_chains`

Specifies to use the exact same number of scan chains as specified by the `dft_min_number_of_scan_chains` attribute when building the scan chain.

`design`

Specifies the name of the top-level design to be checked. You should specify this name in case you have multiple top designs loaded.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-elements` *`element_list`*

Considers only the specified elements for scan chain connection. An element can be a flip-flop, segment, or a hierarchical instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you specify a hierarchical instance, all flops in this hierarchical instance that pass the DFT rule checker and that are mapped to scan for DFT, will be added to the chains.

If some of the scan flops in a hierarchical instance belong to a segment that crosses the boundary of this instance, these scan flops will only be connected if the remaining elements of the segment are also specified with the `-elements` option—either directly or indirectly through another hierarchical instance.

Note: If you specify this option with the `-power_domain` option, the specified elements must belong to the specified power domains.

`-incremental`

Adds new chains in incremental mode, without changing already connected scan chains stored in `/designs/design/dft/report/actual_scan_chains`

Do not use user-defined chains with the same names as the `actual_scan_chains`.

`-include_opcg_segments`

Allows you to include the OPCG segments into the scan chains.

This option must be used when OPCG has been inserted into the design and scan compression is to be added later using the `add_test_compression` command.

`-keep_connected_SE`

Ensures that a flop that is already connected to a shift-enable signal keeps this connection when connected in a scan chain.

If you omit this option, the tool can break the existing connection to the shift enable and connect the shift-enable pin of the flop to either the default shift-enable signal or to the shift-enable signal specified for the chain.

`-pack`

Packs the scan chains to their maximum limit instead of balancing the chains (that is, attempting to create chains with similar lengths).

You can specify a chain-specific constraint using the `-max_length` option of the `define_dft_scan_chain` command or a global constraint by setting the value of the `dft_max_length_of_scan_chains` attribute.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-physical`
: Specifies to use the placement locations of the scan flops to connect the scan chains.

  The placement information is obtained from the DEF file read in with the `read_def` command.

`-power_domain` *`power_domain_list`*
: Considers only the scan flops that belong to the specified power domain(s) for scan chain connection.

`-preview`
: Reports how the scan chains will be connected, but makes no modifications to the netlist. Use this option to verify your scan-chain architecture prior to connecting the scan chains.

`-update_placement`
: Specifies to update the placement of the DFT logic that is added during the scan chain connection process.

  Note: Use this option only in the physical flow after you have already placed the design using `syn_opt -physical`. The option will be ignored if the design is not placed.

`-zipper_stich`
: Attempts to connect all abstract segments defined on the same instance sequentially in a scan chain.

Related Information

Related commands:	write_compression_macro on page 1154
Affected by these constraints:	define_dft abstract_segment on page 915
	define_dft dft_configuration_mode on page 925
	define_dft fixed_segment on page 929
	define_dft floating_segment on page 931
	define_dft preserved_segment on page 963
	define_dft scan_chain on page 966
	define_dft scan_clock_a on page 972
	define_dft scan_clock_b on page 975
	define_dft shift_enable on page 978
	define_dft shift_register_segment on page 981

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	set_compatible_test_clocks on page 1140
Affected by these commands:	check_dft_rules on page 870
	fix_dft_violations on page 1010
	syn_map on page 529
	syn_opt on page 531
Affects this command:	concat_scan_chains on page 897
	report dft_chains on page 1125
Sets these attributes:	Actual Scan Chain attributes
	Actual Scan Segment attributes
Affected by these attributes:	decoded_pin
	dft_lockup_element_type
	dft_max_length_of_scan_chains
	dft_min_number_of_scan_chains
	dft_mix_clock_edges_in_scan_chains
	dft_prefix
	dft_scan_map_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_chains_database

```
create_chains_database
     [-chains actual_scan_chains] [-scandef string]
     [-design design]
```

Creates an internal not-persistent database that pre-processes the scan chain flops location data for the `report_chains_*` commands.

This command needs to be rerun each time the scan chains change.

This command can be used on a placed design that has connected scan chains.

Note: The information in this chains database is not saved in the Genus database.

Options and Arguments

`-design` *design*	Specifies the name of the design for which to create the internal database. If you omit the design name, the top-level design of the current directory of the design hierarchy is used. This option is required if multiple designs are loaded.
`-chains` *actual_scan_chains*	Specifies the names of the actual scan chains for which the scan chain flop location must be preprocessed. By default, the database will be created for all active scan chains.
`-scandef` *string*	Specifies the scanDef file path.

Related Information

Affected by these commands:

report_chains_longest_wires on page 724
report_chains_total_wirelength on page 726
report_chains_wires on page 728

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft

```
define_dft {abstract_segment | boundary_scan_segment
     | dft_configuration_mode | domain_macro_parameters
     | fixed_segment | floating_segment | jtag_macro
     | jtag_instruction | jtag_instruction_register
     | mbist_clock | opcg_domain | opcg_mode
     | opcg_trigger | osc_source
     | preserved_segment | scan_chain
     | scan_clock_a | scan_clock_b | shift_enable
     | shift_register_segment | tap_port | test_bus_port
     | test_clock | test_mode }
```

Defines a DFT object. A DFT object can be a test signal, scan segment, or scan chain.

Options and Arguments

`abstract_segment`	Defines an abstract scan-chain segment object.
`boundary_scan_segment`	Defines a boundary-scan segment object.
`dft_configuration_mode`	Defines a scan mode for DFT configuration purposes.
`domain_macro_parameters`	Defines a set of domain macro parameters used to configure and build the `OPCG` domain macro logic.
`fixed_segment`	Defines a fixed scan-chain segment object.
`floating_segment`	Defines a floating scan-chain segment object.
`jtag_macro`	Defines a pre-instantiated third-party JTAG Macro.
`jtag_instruction`	Defines a user-defined instruction that is serially loaded into a boundary scan device.
`jtag_instruction_register`	Customizes the instruction register to allow adding user-defined instructions.
`mbist_clock`	Defines an MBIST clock object.
`opcg_domain`	Defines an OPCG (clock) domain.
`opcg_mode`	Defines an OPCG mode for ATPG.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`opcg_trigger`	Defines the trigger signal that will enable the trigger macro associated with the specified oscillator source.
`osc_source`	Defines an oscillator source on the output pin of a PLL instance that will drive the OPCG logic.
`pmbist_direct_access`	Defines programmable MBIST direct access interface pins or ports which can be used as an alternative access mechanism for PMBIST.
`preserved_segment`	Defines a preserved scan-chain segment object.
`scan_chain`	Defines a scan chain.
`scan_clock_a`	Defines the scan clock of the master latch for the LSSD scan style.
`scan_clock_b`	Defines the scan clock of the slave latch for the LSSD scan style.
`shift_enable`	Defines a `test_signal` object of type `shift_enable`.
`shift_register_segment`	Defines a shift register scan-chain segment object.
`tap_port`	Defines a TAP port (JTAG signal).
`test_bus_port`	Defines a test bus port.
`test_clock`	Defines a test clock object.
`test_mode`	Defines a `test_signal` object of type `test_mode`

Related Information

Related commands:

define_dft abstract_segment on page 915
define_dft boundary_scan_segment on page 921
define_dft dft_configuration_mode on page 925
define_dft domain_macro_parameters on page 927
define_dft fixed_segment on page 929
define_dft floating_segment on page 931
define_dft jtag_instruction on page 933
define_dft jtag_instruction_register on page 937

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft jtag_macro on page 939
define_dft mbist_clock on page 944
define_dft opcg_domain on page 947
define_dft opcg_mode on page 951
define_dft opcg_trigger on page 953
define_dft osc_source on page 956
define_dft pmbist_direct_access on page 958
define_dft preserved_segment on page 963
define_dft scan_chain on page 966
define_dft scan_clock_a on page 972
define_dft scan_clock_b on page 975
define_dft shift_enable on page 978
define_dft shift_register_segment on page 981
define_dft tap_port on page 983
define_dft test_bus_port on page 985
define_dft test_clock on page 990
define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft abstract_segment

```
define_dft abstract_segment [-name segment_name]
     {-module subdesign|-instance instance|-libcell cell}
     -sdi subport [-inversion] -sdo subport [-tail_inversion]
     -clock_port subport [-rise|-fall] [-off_state {high|low}]
     [ -tail_clock_port subport
       [-tail_edge_rise | -tail_edge_fall]
       [-tail_clock_off_state {high|low}] ]
     [-other_clock_port subport
       [-other_clock_edge {rise|fall}]]...
     { { -shift_enable_port subport -active {high|low}
       | -connected_shift_enable }
     | { -scan_clock_a_port subport -scan_clock_b_port subport
       | -connected_scan_clock_a -connected_scan_clock_b } }
     [ -test_mode_port subport
       -test_mode_active {low|high} ]...
     -length integer
     [-skew_safe {-skew_flop|-skew_latch}] [-head_skew_safe]
     [-dft_configuration_mode dft_config_mode_name]
```

Defines an abstract segment. An abstract segment can be defined for objects of type blackbox, logic abstract module, or libcell timing model.

An abstract segment is a user-specified scan segment used at the next level of integration to define the sets of scan chains previously created for the object.

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft abstract_segment` constraints in:

`/designs/top_design/dft/scan_segments`

Options and Arguments

`-active {low|high}` Specifies the active value for the shift-enable port.

`-clock_port subport`

Specifies the clock port—at the boundary of the blackbox or logic abstract module—driving the flip-flops at the head of the segment.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-connected_scan_clock_a (-connected_scan_clock_b)`

Indicates that the `scan_clock_a` (`scan_clock_b`) port of the module boundary is driven by external logic (preconnected). The external logic connected to the `scan_clock_a` (`scan_clock_b)` pin of the module will not be modified by the scan configuration engine.

Note: This option applies only for the clocked LSSD scan style.

`-connected_shift_enable`

Indicates that the shift enable port of the module boundary is driven by external logic (preconnected) or that the shift enable signal is internally generated within the module boundary. In either case, the external logic connected to the shift enable pin of the module, or the internal logic driving the shift enable pins of the flip-flops in the module will not be modified by the scan configuration engine.

This option cannot be specified together with the `-shift_enable` option.

`-dft_configuration_mode` *`dft_configuration_mode_name`*

Specifies in which scan mode the abstract segment will be connected in the scan chains.

`-head_skew_safe` Indicates whether the abstract segment has a data lockup element connected at the head of its scan chain.

`-instance` *`instance`* Specifies the instance name of the module for which the abstract segment is defined.

`-inversion` Indicates an inversion at the scan data input pin of the abstract segment.

`-length` *`integer`* Specifies the length of the abstract segment.

`-libcell` *`cell`* Specifies the library cell for which the abstract segment is defined. This option applies to library cells that are implemented as timing models and whose description includes the relevant test-related pins (such as scan data input and output, clock, shift-enable) to infer the scan chain architecture.

`-module` *`subdesign`* Specifies the subdesign (module) to which the element belongs.

`-name` *`segment_name`* Defines a name for the segment that you can use to reference in the `define dft scan chain` constraint.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-off_state {high|low}`

Specifies the off state of the system clock specified through the `-clock_port` option.

Note: This option applies only to the clocked LSSD scan style.

`-other_clock_edge {rise | fall}`

Specifies the active edge of the clock specified through the `-other_clock_port` option.

Default: `rise`

`-other_clock_port` *`subport`*

Specifies the clock port—at the boundary of the blackbox or logic abstract module—of another system clock used to drive some flip-flops in the abstract segment.

This option is only required if the clock is different from the clock specified through the `-clock_port` option or the `-tail_clock_port` option.

`{-rise | -fall}`

Specifies the active edge of the clock specified through the `-clock_port` option.

Default: `-rise`

`-scan_clock_a_port (-scan_clock_b_port)` *`subport`*

Specifies the `scan_clock_a` (`scan_clock_b`) port at the boundary of the blackbox or logic abstract module to which the segment belongs. Specify this option to have the `connect_scan_chains` command make the connection from the top-level `scan_clock_a` (`scan_clock_b`) signals to the `scan_clock_a` (`scan_clock_b)` port of the module.

Note: This option applies only for the clocked LSSD scan style.

`-sdi (-sdo)`

Specifies the scan data input (scan data output) of the segment.

- For a segment in a blackbox, specify a subport (port of the blackbox or logic abstract module).
- For a segment defined for a libcell, specify a pin of the libcell.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-shift_enable_port` *`subport`*

Specifies the shift enable port at the boundary of the blackbox or logic abstract module to which the segment belongs. Specify this option if you want the `connect_scan_chains` command to make the connection from the top-level shift-enable signals to the shift-enable ports of the modules.

This option cannot be specified together with the `-connected_shift_enable` option.

`-skew_flop`

Specifies an edge-triggered flop has been inserted as the terminal lockup element for the skew-safe segment.

This option must be specified with the `-skew_safe` option.

`-skew_latch`

Specifies a level-sensitive latch has been inserted as the terminal lockup element for the skew-safe segment.

This option must be specified with the `-skew_safe` option.

`-skew_safe`

Indicates whether the abstract segment has a data lockup element connected at the end of its scan chain.

`-tail_clock_off_state {high|low}`

Specifies the off state of the system clock specified through the `-tail_clock_port` option.

Note: This option applies only to the clocked LSSD scan style.

`-tail_clock_port` *`port`*

Specifies the clock port—at the boundary of the blackbox or logic abstract module—driving the flip-flops at the tail of the segment. This option is only required if the clock used at the tail of the abstract segment is different from the clock specified through the `-clock_port` option.

`-tail_edge_rise | -tail_edge_fall`

Specifies the active edge of the clock specified through the `-tail_clock_port` option.

Default: `-tail_edge_rise`

`-tail_inversion`

Indicates an inversion at the scan data output pin of the abstract segment.

`-test_mode_active {low | high}`

Specifies the active value for the test-mode port.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This option should immediately follow the corresponding `-test_mode_port` option.

`-test_mode_port` *`subport`*

Specifies the test mode port at the boundary of the blackbox module to which the segment belongs.

Specify this option when the block-level design includes test-mode activated logic.

When the test-mode signals are specified, the propagated values of the top-level test-mode signals must match the expected block-level test-mode values and the segment must also pass the clock-controllability rule checks in order for the segment to be included into a top-level scan chain.

Note: The tool does not make connections to the test-mode ports of the block-level design. The connections should already exist in the netlist.

Examples

- The following example defines an abstract segment with length 3 in blackbox module `b`. The clock driving the flip-flops at the tail of the segment is the same as the clock driving the first elements in the segment.

```
legacy_genus:/> define_dft abstract_segment -name a1 -module b -length 3 \
-sdi {p4[0]} -sdo {p5[0]} -shift_enable {p6[0]} -active high -clock p3 -rise
```

- The following example defines an abstract segment in a timing model reference `COMBELEM` with length 20.

```
legacy_genus:/> define_dft abstract_segment -name combElem_seg \
-libcell COMBELEM -sdi A -sdo Z -shift_enable_port B -active hi \
-clock_port D2 -rise -length 20
```

- The following example defines an abstract segment `ABS` with length 10 in instance `o1`. The same clock `clk` with active rising edge is used for all flip-flops of the segment.

```
define_dft abstract_segment -instance o1  -sdi sdi -sdo sdo \
-shift_enable_port se -active hi -clock_port clk -rise -length 10 -name ABS
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this constraint:	define_dft dft_configuration_mode on page 925
	define_dft scan_chain on page 966
Affects these commands:	check_dft_rules on page 870
	connect_scan_chains on page 906
	fix_scan_path_inversions on page 1014
	report dft_chains on page 1125
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft boundary_scan_segment

```
define_dft boundary_scan_segment [-name segment_name]
     {-module subdesign | -instance instance | -libcell cell}
     {-bsdl_string string | -bsdl_file file}
     [-differential_pair {positive_leg_pin negative_leg_pin}]
     [-mode_a mode_a_pin]... [-mode_b mode_b_pin]...
     [-mode_c mode_c_pin]... [-highz highz_pin]
     -tdi tdi_pin -tdo tdo_pin [-clockdr clockdr_pin]
     [-capturedr capturedr_pin]
     [-updatedr updatedr_pin]
     [-shiftdr shiftdr_pin] [-index bsr_position]
     [-acdcsel_11496 pin][-acpclk_11496 pin]
     [-acpsen_11496 pin] [-acptrenbl_11496 pin]
     [-acpulse_11496 pin]
```

Defines a boundary scan segment with its associated pins for connection along the TDI-TDO path in the boundary scan register, connection to the `JTAG_Macro`, and optionally defines its position in the boundary scan register.

You can define multiple boundary scan segments on the same instance, as long as the `TDI` and `TDO` pins of the segments are unique.

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft boundary_scan_segment` command in:

`/designs/top_design/dft/boundary_scan_segments`

Options and Arguments

`-acdcsel_11496 pin`

Specifies the name of the pin on the boundary scan segment that will be driven by the `JTAG_ACDCSEL` test signal.

`-acpclk_11496 pin`

Specifies the name of the pin on the boundary scan segment that will be driven by the `JTAG_ACPSCLK` test signal.

`-acpsen_11496 pin`

Specifies the name of the pin on the boundary scan segment that will be driven by the `JTAG_ACPSEN` test signal.

`-acptrenbl_11496 pin`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the name of the pin on the boundary scan segment that will be driven by the `JTAG_ACTRENBL` test signal.

`-acpulse_11496` *`pin`*

Specifies the name of the pin on the boundary scan segment that will be driven by the `JTAG_ACPULSE` test signal.

`-bsdl_file` *`file`*

Specifies the file containing the BSDL abstract string for the boundary scan segment.

`-bsdl_string` *`string`*

Specifies the BSDL abstract string for the boundary scan segment.

`-capturedr` *`capturedr_pin`*

Specifies the name of the `CAPTURE_DR` pin on the boundary scan segment.

`-clockdr` *`clockdr_pin`*

Specifies the name of the `CLOCK_DR` pin on the boundary scan segment.

`-differential_pair` {*`positive_leg_pin negative_leg_pin`*}

Specifies the differential pair in the form of a positive leg pin name and negative leg pin name on the boundary scan segment.

`-highz` *`highz_pin`*

Specifies the name of the `HIGHZ` pin on the boundary scan segment

`-index` *`bsr_position`*

Specifies the relative position of the boundary scan segment in the BSR. Specify an integer value of zero or greater.

`-instance` *`instance`*

Specifies the instance name of the module for which the boundary scan segment is defined.

`-libcell` *`cell`*

Specifies the library cell for which the boundary scan segment is defined. This option applies to library cells that are implemented as timing models and whose description includes the `JTAG_Macro` related pins specified on the command line when defining the boundary scan segment.

`-mode_a` *`mode_a_pin`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the name of the `MODE_A` pin on the boundary scan segment

`-mode_b` *`mode_b_pin`*

Specifies the name of the `MODE_B` pin on the boundary scan segment

`-mode_c` *`mode_c_pin`*

Specifies the name of the `MODE_C` pin on the boundary scan segment

`-module` *`subdesign`* Specifies the subdesign (module) to which the element belongs.

`-name` *`segment_name`* Defines a name for the boundary scan segment.

`-shiftdr` *`shiftdr_pin`*

Specifies the name of the `SHIFT_DR` pin on the boundary scan segment.

`-tdi` *`tdi_pin`* Specifies the name of the `TDI` pin on the boundary scan segment.

`-tdo` *`tdo_pin`* Specifies the name of the `TDO` pin on the boundary scan segment.

`-updatedr` *`updatedr_pin`*

Specifies the name of the `UPDATE_DR` pin on the boundary scan segment.

Example

- The following example defines an boundary scan segment using a set of differential port pairs.

```
legacy_genus:/> define_dft boundary_scan_segment -instance i_pads \
-bsdl_file pads_bcell.abstract -mode_a MODE_A -mode_b MODE_B -mode_c \
MODE_C -highz HIGHZ -tdi TDI -tdo TDO -clockdr CLOCKDR -updatedr UPDATEDR\
-shiftdr SHIFTDR -index 4 \
-differential_pair {in1 in2} \
-differential_pair {out1 out2}
```

- The following example defines a boundary scan segment with three `mode_a`, two `mode_b`, and one `mode_c` pins where:
  - Each `mode_a` pin on the boundary-scan segment will be connected to the `JTAG_MACRO` `JTAG_INSTRUCTION_DECODE_MODE_A` output pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Each `mode_b` pin on the boundary-scan segment will be connected to the `JTAG_MACRO JTAG_INSTRUCTION_DECODE_MODE_B` output pin.
- Each `mode_c` pin on the boundary-scan segment will be connected to the `JTAG_MACRO JTAG_INSTRUCTION_DECODE_MODE_C` output pin.

```
legacy_genus:/> define_dft boundary_scan_segment-instance i_pads \
-bsdl_file pads_bcell.abstract -mode_a MODE_A1 -mode_a MODE_A2 \
-mode_a MODE_A3 -mode_b MODE_B1 -mode_b MODE_B2 -mode_c MODE_C \
-highz HIGHZ -tdi TDI -tdo TDO -clockdr CLOCKDR \
-updatedr UPDATEDR -shiftdr SHIFTDR -index 4 -differential_pair {in1 in2}\
-differential_pair {out1 out2}
```

Related Information

Affects these commands:	insert_dft boundary_scan on page 1026
	write_bsdl on page 1151
Sets these attributes:	bcell_segment
	differential

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft dft_configuration_mode

```
define_dft dft_configuration_mode
     [-name scan_mode_name]
     [-mode_enable_high test_signal ...]
     [-mode_enable_low test_signal...]
     [-jtag_instruction jtag_instruction]
     [-type {scan |
       wrapper [-usage { extest | intest | mission}] }
     [design]
```

Defines a scan mode. Scan modes can be used to build the top-level scan chains with specific elements in different modes of operation (multi-mode, 1500 wrapper insertion), or when concatenating default scan chains into a single longer scan chain in a different mode of operation.

You can find the objects created by the `define_dft dft_configuration_mode` command in:

`/designs/top_design/dft/dft_configuration_modes`

Options and Arguments

design
: Specifies the name of the top-level design for which the scan mode is defined. Specify this name if you have multiple top designs loaded.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-jtag_instruction` *jtag_instruction_name*
: Specifies the JTAG instruction which controls the scan chains in the current mode.

`-mode_enable_high` *test_signal...*
: Name of the test signal(s) set to a `logic_1` value that controls scan chains in the current mode.

`-mode_enable_low` *test_signal...*
: Name of the test signal(s) set to a `logic_0` value that controls the scan chains in the current mode.

`-name`
: Name of the scan mode.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-type {scan | wrapper}`

Specifies the type of the scan mode.

`-usage {extest | intest | mission}`

Specifies the usage of the scan mode for IEEE 1500 core wrapper insertion.

This option is only valid with `-type wrapper`.

Example

The following example defines scan mode `scanModeA`. Test signal `test1` is specified to have an active high logic value and test signal `test2` is specified to have an active low logic value in this mode of operation:

```
define_dft dft_configuration_mode -name scanModeA design \
    -mode_enable_high test1 -mode_enable_low test2
```

Related Information

Affects these commands:	report dft_chains on page 610
	check_dft_rules on page 870
	concat_scan_chains on page 897
	connect_scan_chains on page 906
	define_dft abstract_segment on page 915
	write_atpg on page 1148
	write_dft_abstract_model on page 1162
	write_dft_deterministic_test_points on page 1169
	write_et_atpg on page 1181
	write_et_bsv on page 1188
	write_scandef on page 1216
Affects these attributes:	DFT Configuration Mode Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft domain_macro_parameters

```
define_dft domain_macro_parameters [-name name]
     [-max_num_pulses integer]
     [-counter_length integer | -max_trigger_delay float]
     [-min_target_period float] [-design design]
```

Defines a set of domain macro parameters used to configure and build the `OPCG` domain macro logic.

You must specify the command with either `-counter_length` or `-max_trigger_delay`.

The command returns the directory path to the `domain_macro_parameter` object that it creates.You can find the objects created by the `define_dft domain_macro_parameters` constraints in:

```
/designs/top/dft/opcg/domain_macro_parameters/
```

Options and Arguments

`-counter_length` *`integer`*

Specifies the number of bits in the down counter register.

Specify a number between 4 and 7, or specify 0 if no counter should be inserted.

`-design` *`design`*

Specifies the name of the design for which the domain macro parameter set is defined.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

This option is required if multiple designs are loaded.

`-max_num_pulses` *`integer`*

Specifies the number of high speed pulses the domain macro should be able to generate. The maximum number you can specify is 8.

Default: `2`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max_trigger_delay` *`float`*

Specifies the time (in picoseconds) after which the first pulse must be issued by the domain macro when it is triggered by the enabled by the `TRIGGERRUN` output signal generated by the trigger macro.

`-min_target_period` *`float`*

Specifies the minimum target period (in picoseconds) at which the domain macro must operate.

Note: This option is required when inserting OPCG logic into a technology mapped design. It is used to specify the `opcg_pllclk` period in the tool-generated SDC constraints file used to synthesize the domain macro logic.

Default: `1000`

`-name` *`name`*

Specifies the `domain_macro_parameter` object name of the domain macro parameter set.

Default: `DOMAIN_MACRO_PARAMETER_n`

Related Information

Affects this constraint:	define_dft opcg_domain on page 947
Affects this command:	insert_dft opcg on page 1063
Sets these attributes:	Domain Macro Parameters Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft fixed_segment

```
define_dft fixed_segment [-name segment_name]
     {pin|port|instance|segment_name} ...
```

Defines a fixed segment. In a fixed segment, the elements will be connected in the specified order during scan chain connection; they cannot be reordered by a physical scan reordering tool.

A fixed segment is a user-specified scan segment which can be associated with either

- A user-defined top-level chain—created using the `define_dft scan_chain` command
- A tool-created scan chain—created using the `connect_scan_chains` command

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft fixed_segment` constraints in:

`/designs/top_design/dft/scan_segments`

Options and Arguments

`{pin|port|instance|segment_name}`	Specifies an element in the scan segment being defined. List the elements in the order they should appear in the scan chain (in shift order, that is, left-most corresponds to first bit shifted-in). An element can be hierarchical pin, a port, a flip-flop instance, a combinational gate, or a scan segment. **Note:** If the segment goes through a multi-input/output combinational gate, you must indicate the scan path through the gate by specifying its input and output pin as two separate consecutive elements.
`-name segment_name`	Defines a name for the segment that you can use to reference in the `define_dft scan_chain` constraint.

Examples

- The following example defines the fixed segment used in the example for define_dft scan_chain on page 966.

  ```
  define_dft fixed_segment -name segBody *seq/out_reg_1 *seq/out_reg_3
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example defines a fixed segment that contains two combinational components, four sequential registers, a scan abstract segment, and a combinational component endpoint.

```
define_dft fixed_segment -name fixedSeg \
    i_core/i_anor1/B0 i_core/i_anor1/Y i_core/i_anor2/B0 i_core/i_anor2/Y \
    i_core/i_flop11 i_core/i_flop22 i_core/i_flop33 i_core/i_flop44 \
    absSeg \
    i_core/bufToAnchorSeg/A i_core/bufToAnchorSeg/Y
```

Related Information

Affects this constraint:	define_dft scan_chain on page 966
Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft floating_segment

```
define_dft floating_segment [-name segment_name]
     {pin|port|instance|segment_name} ...
```

Defines a floating segment. In a floating segment, the order of the elements can be changed during scan configuration and by a physical scan reordering tool.

A floating segment is a user-specified scan segment which can be associated with either

- A user-defined top-level chain—created using the `define_dft scan_chain` command
- A tool-created scan chain—created using the `connect_scan_chains` command

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft floating_segment` constraints in:

`/designs/top_design/dft/scan_segments`

Options and Arguments

`-name segment_name`	Defines a name for the segment that you can use to reference in the `define_dft scan_chain` constraint.
`{pin|port|instance|segment_name}`	Specifies an element in the scan segment being defined. List the elements in the order they should appear in the scan chain (in shift order, that is, left-most corresponds to first bit shifted-in). An element can be hierarchical pin, a port, a flip-flop instance, or a scan segment.

Examples

- The following example defines the floating segments used in the example for define_dft scan_chain on page 966.

```
legacy_genus:/designs/test> define_dft floating_segment -name segHead \
*seq/out_reg_4 *seq/out_reg_5
legacy_genus:/designs/test> define_dft floating_segment -name segTail \
*seq/out_reg_0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this constraint:	define_dft scan_chain on page 966
Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft jtag_instruction

```
define_dft jtag_instruction -name string -opcode string
     [-register string] [-length integer]
     [-register_tdi {pin|port}]
     [-register_tdo {pin|port}]
     [-register_shiftdr {pin|port}]
     [-register_shiftdr_inverted]
     [-register_shiftdr_state {pin|port}]
     [-register_reset_inverted]
     [-register_capturedr {pin|port}]
     [-register_capturedr_state {pin|port}]
     [-register_clockdr {pin|port}]
     [-register_updatedr {pin|port}]
     [-register_updatedr_state {pin|port}]
     [-register_tck {pin|port}]
     [-register_reset {pin|port}]
     [-register_runidle {pin|port}]
     [-register_decode {pin|port}]
     [-capture string]
     [-tap_tdo {pin|port}] [-tap_decode {pin|port}]
     [-private] [-design design]
```

Defines a user-defined instruction that is serially loaded into a boundary scan device.

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft jtag_instruction` in:

`/designs/top_design/dft/boundary_scan/jtag_instructions/instruction`

Options and Arguments

`-capture string`	Specifies the values that must be captured into a register during the `CaptureDR` state.
`-design design`	Specifies the name of the design for which the JTAG instruction is defined. If you omit the design name and multiple designs are loaded, the top-level design of the current directory of the design hierarchy is used.
`-length integer`	Specifies the length of the custom test data register (TDR).
`-name string`	Specifies the name of the user-defined instruction.
`-opcode string`	Specifies the binary code for this instruction.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-private` Specifies that the defined instruction is private.

`-register` *`string`* Specifies the name of the custom test data register (TDR).

`-register_capturedr {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_CAPTUREDR` pin on the `JTAG_MACRO` subdesign.

`-register_capturedr_state {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_CAPTUREDR_STATE` pin on the `JTAG_MACRO` subdesign.

`-register_clockdr {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_CLOCKDR` pin on the `JTAG_MACRO` subdesign.

`-register_decode {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_INSTRUCTION_DECODE_`*`instruction`* pin on the `JTAG_MACRO` subdesign, where *`instruction`* is the name that you specified through the `-name` option.

`-register_reset {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_RESET` pin on the `JTAG_MACRO` subdesign.

`-register_reset_inverted`

Specifies that the `JTAG_RESET` pin of the custom test register (TDR) has an active low polarity.

`-register_runidle {`*`pin`*`|`*`port`*`}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_RUNIDLE` pin on the `JTAG_MACRO` subdesign.

`-register_shiftdr {`*`pin`*`|`*`port`*`}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_SHIFTDR` pin on the `JTAG_MACRO` subdesign.

`-register_shiftdr_state {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_SHIFTDR_STATE` pin on the `JTAG_MACRO` subdesign.

`-register_shiftdr_inverted`

Specifies that the `JTAG_SHIFTDR` pin of the custom test register (TDR) has an active low polarity.

`-register_tck {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_TCK` pin on the `JTAG_MACRO` subdesign.

`-register_tdi {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_TDI` pin on the `JTAG_MACRO` subdesign.

`-register_tdo {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_register_TDO` pin on the `JTAG_MACRO` subdesign, where *`register`* is the name that you specified through the `-register` option

`-register_updatedr {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_UPDATEDR` pin on the `JTAG_MACRO` subdesign.

`-register_updatedr_state {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the `JTAG_UPDATEDR_STATE` pin on the `JTAG_MACRO` subdesign.

`-tap_decode {pin|port}`

Specifies the name of the instruction-specific decode pin that must be created on the `JTAG_MACRO` subdesign.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-tap_tdo {pin|port}`

Specifies the name of the instruction-specific test data output (TDO) pin that must be created on the `JTAG_MACRO` subdesign.

Example

- The following example defines a private instruction `PROGRAM_TCB` for custom test data register `TCB_REG` that has a length of 8 bits. The opcode for the instruction is `0101`.

```
define_dft jtag_instruction -name PROGRAM_TCB -opcode 0101 -register TCB_REG \
 -length 8 -private
```

Related Information

Affects these commands:	insert_dft boundary_scan on page 1026
	insert_dft jtag_macro on page 1046
Related command:	define_dft jtag_instruction_register on page 937
Sets these attributes:	JTAG Instruction Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft jtag_instruction_register

```
define_dft jtag_instruction_register
     -name string
     [-length integer] [-capture string]
     [-design design]
```

Customizes the instruction register to allow adding user-defined instructions.

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft jtag_instruction_register` in:

`/designs/top_design/dft/boundary_scan/register_name`

Options and Arguments

`-capture string`	Specifies the capture value of the instruction register. According to the IEEE 1149.1 standard the last two bits of the capture value must be `01`. *Default*: 01
`-design design`	Specifies the name of the design for which the instruction register is customized. If you omit the design name and multiple designs are loaded, the top-level design of the current directory of the design hierarchy is used.
`-length integer`	Specifies the length of the instruction register. The length of the register is determined by the number of user-defined instructions that you want to add. If n is the number of bits in the instruction register, a total of 2^n instructions can be defined including the four mandatory instructions. *Default*: 2
`-name string`	Specifies the name of the user-defined instruction register.

Examples

- The following example defines instruction register `INSTR_REGISTER` with length 3. A register of length 3 allows you to create 2^3 instructions, or four user-defined instructions besides the four mandatory instructions.

  ```
  define_dft jtag_instruction_register -name INSTR_REGISTER -length 3
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related commands:	define_dft jtag_instruction on page 933
Affects these commands:	insert_dft boundary_scan on page 1026
	insert_dft jtag_macro on page 1046
Related attributes:	JTAG Instruction Register Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft jtag_macro

```
define_dft jtag_macro
     [-module subdesign] [-libcell libcell]
     [-instance instance]
     [-bsr_shiftdr {pin|port|subport}]
     [-bsr_clockdr {pin|port|subport}]
     [-bsr_updatedr {pin|port|subport}]
     [-reset {pin|port|subport}]
     [-runidle {pin|port|subport}]
     [-clockdr {pin|port|subport}]
     [-shiftdr {pin|port|subport}]
     [-updatedr {pin|port|subport}]
     [-capturedr {pin|port|subport}]
     [-shiftdr_state {pin|port|subport}]
     [-updatedr_state {pin|port|subport}]
     [-capturedr_state {pin|port|subport}]
     [-exit1dr_state {pin|port|subport}]
     [-mode_a {pin|port|subport}]
     [-mode_b {pin|port|subport}]
     [-mode_c {pin|port|subport}]
     -tdi {pin|port|subport} -tdo {pin|port|subport}
     -tck {pin|port|subport} -tms {pin|port|subport}
     [-trst pin|port|subport]
     [-boundary_tdo {pin|port|subport}]
     [-tdo_enable {pin|port|subport}]
     [-highz {pin|port|subport}]
     [-dot6_acdcsel {pin|port|subport}]
     [-dot6_acpulse {pin|port|subport}]
     [-dot6_preset_clock {pin|port|subport}]
     [-dot6_trcell_enable {pin|port|subport}]
     [-por {pin|port|subport}] [-name string]
```

Identifies a pre-instantiated JTAG Macro.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-boundary_tdo {pin|port|subport}`

Specifies the boundary register `TDO output` pin on the JTAG macro.

`-bsr_clockdr {pin|port|subport}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the clock data register (CLOCKDR) output pin for the boundary-scan register.

-bsr_shiftdr {*pin*|*port*|*subport*}

Specifies the shift data register (SHIFTDR) output pin for the boundary-scan register.

-bsr_updatedr {*pin*|*port*|*subport*}

Specifies the update data register (UPDATEDR) output pin for the boundary-scan register.

-clockdr {*pin*|*port*|*subport*}

Specifies the clock data register (CLOCKDR) output pin for the custom test data register.

-capturedr {*pin*|*port*|*subport*}

Specifies the capture data register (CAPTUREDR) output pin for the custom test data register.

-capturedr_state {*pin*|*port*|*subport*}

Specifies the capture data register (CAPTUREDR_STATE) output pin for the custom test data register.

-dot6_acdcsel {*pin*|*port*|*subport*}

Specifies the logical OR of the decoded EXTEST_PULSE and EXTEST_TRAIN instructions. The tool connects this signal to the AC Mode pin of all test receivers when it inserts the boundary scan logic. This signal also controls the multiplexer which is added to the output boundary scan cells.

-dot6_acpulse {*pin*|*port*|*subport*}

Specifies the AC test signal output of the JTAG macro.

-dot6_preset_clock {*pin*|*port*|*subport*}

Specifies the preset_clock output pin that provides a positive-active edge-sensitive clock signal to test receivers that have edge-sensitive initialization.

-dot6_trcell_enable {*pin*|*port*|*subport*}

Specifies the logical OR of EXTEST, EXTEST_PULSE and EXTEST_TRAIN used to enable the test receiver cells.

-exit1dr_state {*pin*|*port*|*subport*}

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the exit1dr data register (EXIT1DR) state output pin on the JTAG macro.

`-highz {pin|port|subport}`

Specifies the `HIGHZ` output pin to place the I/O pads in their `HIGHZ` state.

`-instance instance` Specifies the path name of the `JTAG_MACRO` instance.

`-libcell cell` Specifies the library cell for the `JTAG_MACRO` instance.

`-mode_a {pin|port|subport}`

Specifies the `mode_a` output pin to configure boundary cells in the boundary-scan register.

`-mode_b {pin|port|subport}`

Specifies the `mode_b` output pin to configure boundary cells in the boundary-scan register.

`-mode_c {pin|port|subport}`

Specifies the `mode_c` output pin to configure boundary cells in the boundary-scan register.

`-module subdesign` Specifies the subdesign (module) name of the `JTAG_Macro` instance.

`-name subdesign` Specifies the name of the `JTAG_MACRO`. If not specified, an object will be added under the `boundary_scan/jtag_macros` vdir with a default name of `jtag_macro_n`.

`-por {pin|port|subport}`

Specifies the power-on reset input pin on the JTAG macro.

`-reset {pin|port|subport}`

Specifies the reset output pin indicating that the JTAG macro is in the Test-Logic-Reset state.

`-runidle {pin|port|subport}`

Specifies the `JTAG_RUNIDLE` output pin indicating that the JTAG macro is in the Run-Test-Idle state.

`-shiftdr {pin|port|subport}`

Specifies the shift data register (`SHIFTDR`) output pin for the custom test data register.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-shiftdr_state {pin|port|subport}`

Specifies the shift data register (`SHIFTDR_STATE`) output pin for the custom test data register.

`-tck {pin|port|subport}`

Specifies the TAP controller `TCK` input pin on the JTAG macro.

`-tdi {pin|port|subport}`

Specifies the TAP controller `TDI` input pin on the JTAG macro.

`-tdo {pin|port|subport}`

Specifies the TAP controller `TDO` output pin on the JTAG macro.

`-tdo_enable {pin|port|subport}`

Specifies the enable output pin which drives the JTAG `TDO` output enable pin.

`-tms {pin|port|subport}`

Specifies the TAP controller `TMS` input pin on the JTAG macro.

`-trst {pin|port|subport}`

Specifies the TAP controller `TRST` input pin on the JTAG macro.

`-updatedr {pin|port|subport}`

Specifies the update data register (`UPDATEDR`) output pin for the custom test data register.

`-updatedr_state {pin|port|subport}`

Specifies the update data register (`UPDATEDR_STATE`) output pin for the custom test data register.

Examples

- The following example defines a third party TAP controller with a specified instance location.

```
legacy_genus:/> define_dft jtag_macro -instance user_defined_jtag \
-highz MY_JTAG_INSTRUCTION_DECODE_CTRL_HIGHZ \
-bsr_clockdr MY_JTAG_BOUNDARY_CLOCKDR -bsr_shiftdr MY_JTAG_BOUNDARY_SHIFTDR \
-bsr_updatedr MY_JTAG_BOUNDARY_UPDATEDR \
-mode_a  MY_JTAG_INSTRUCTION_DECODE_MODE_A \
-mode_b MY_JTAG_INSTRUCTION_DECODE_MODE_B \
-mode_c MY_JTAG_INSTRUCTION_DECODE_MODE_C -tdi MY_JTAG_TDI -tdo MY_JTAG_TDO \
-tms MY_JTAG_TMS -tck MY_JTAG_TCK -trst MY_JTAG_TRST \
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
-tdo_enable MY_JTAG_ENABLE_TDO -boundary_tdo MY_JTAG_BOUNDARY_TDO \
-por MY_JTAG_POR -name JM1
```

- The following example defines a third party TAP controller without a `TRST` input pin.

```
legacy_genus:/> define_dft jtag_macro -module MY_JTAG_MACRO \
-highz MY_JTAG_INSTRUCTION_DECODE_CTRL_HIGHZ \
-bsr_clockdr MY_JTAG_BOUNDARY_CLOCKDR -bsr_shiftdr MY_JTAG_BOUNDARY_SHIFTDR \
-bsr_updatedr MY_JTAG_BOUNDARY_UPDATEDR \
-mode_a MY_JTAG_INSTRUCTION_DECODE_MODE_A \
-mode_b MY_JTAG_INSTRUCTION_DECODE_MODE_B \
-mode_c MY_JTAG_INSTRUCTION_DECODE_MODE_C -tdi MY_JTAG_TDI -tdo MY_JTAG_TDO \
-tms MY_JTAG_TMS -tck MY_JTAG_TCK -tdo_enable MY_JTAG_ENABLE_TDO \
-boundary_tdo MY_JTAG_BOUNDARY_TDO -por MY_JTAG_POR -name JM1
```

Related Information

Sets these attributes:	JTAG Macro Attributes
Related command:	insert_dft jtag_macro on page 1046

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft mbist_clock

```
define_dft mbist_clock -name mbist_clock
     [-design design] [-internal_clock_source]
     -period integer
     [-domain domain]
     [-hookup_pin pin ] [-hookup_period integer]
     [-hookup_polarity {non_inverted | inverted}]
     [-is_jtag_tck] [is_srclk] port
```

Defines an MBIST (PMBIST) clock and associates a clock waveform with the clock. The clock waveform can be different from the system clocks. You must define all MBIST (PMBIST) clocks that are referenced in the MBIST (PMBIST) configuration file to enable the DFT rules checking associated with MBIST (PMBIST) insertion in the design.

Note: The system function for the boundary cell associated with the external clock source must be `clock`.

The command returns the virtual directory path to the `mbist` object that it creates.You can find the objects created by the `define_dft mbist_clock` constraints in:

`/designs/design/dft/mbist/mbist_clocks`

Options and Arguments

`-design` *design*	Specifies the name of the design for which the MBIST (PMBIST) clock is defined. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
`-domain` *domain*	Specifies the PMBIST clock domain to which the defined PMBIST clock belongs. *Default*: PMBIST clock name.
`-hookup_period` *integer*	Specifies the period interval at the hookup pin. The hookup period is specified in picoseconds. *Default*: value of `-period` option
`-hookup_pin` *pin*	Specifies the core-side hookup pin to be used by the `insert_dft mbist` command to make the MBIST (PMBIST) clock connection.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hookup_polarity {non_inverted | inverted}`

Specifies the polarity of the MBIST (PMBIST) clock signal at the core-side hookup pin relative to the specified port.

Default: `non_inverted`

`-internal_clock_source`

Specifies that the MBIST (PMBIST) clock is either driven by a clock source internal to the design or by an external clock source, design port, that has a free-running clock applied at the tester.

`-is_jtag_tck`

Specifies that the MBIST (PMBIST) clock is defined as a JTAG clock.

`-is_srclk`

Specifies that the MBIST (PMBIST) clock will be used as repair clock for all memories (which require repair).

Note: If no clock is defined as repair clock, the repair clock of a memory will be the same as the one used for running the memory test for that particular memory.

`-name` *`mbist_clock`*

Specifies the name of the MBIST (PMBIST) clock that is being defined.

Each clock object in your design must have a unique name. If you define a new MBIST (PMBIST) clock with the same name as an existing clock, an error message will be issued.

Note: The clock name is referenced within the MBIST (PMBIST) configuration file to access this clock object. The clock name also allows you to search for the clock later (through the `find` command) or to recognize it in reports.

`-period` *`integer`*

Specifies the clock period interval at the specified port. The clock period is specified in picoseconds.

Default: `50000` (20 MHz test clock)

`port`

Specifies the MBIST (PMBIST) clock input port, or the *test time control* port in case of an internal MBIST (PMBIST) clock.

The test time control port is any port on the design which can toggle during memory test to control the number of test cycles executed to ensure the internally clocked MBIST (PMBIST) test completes.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example defines three MBIST clocks, the second one needing a hookup pin to avoid a PLL and the third one specifying an internal clock source.

```
define_dft mbist_clock -name CLK1X -period 20000 CLK1
define_dft mbist_clock -name CLK2X -period 20000 -hookup_pin PLLOUTA  \
    -hookup_period 10000 CLK2
define_dft mbist_clock -name CLK3I -period 30000 -hookup_pin OSCOUT \
    -internal_clock_source CLK2
```

  For the MBIST clock object `CLK2X` there is a clock frequency multiplication factor of 2 through the PLL and it is in phase with the port CLK2. For the MBIST clock object `CLK3I` the clock is driven by an on-chip oscillator which reuses design port `CLK2` as the test time control port.

Related Information

Affects this command:	insert_dft pmbist on page 1065
Sets these attributes:	MBIST Clock Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft opcg_domain

```
define_dft opcg_domain [-name name]
     -osc_source osc_source
     { [-domain_macro_parameter domain_macro_parameter]
       [-location {subport | port | pin}]
     | [-counter_length integer] [-max_num_pulses integer]}
     -opcg_trigger opcg_trigger
     -min_domain_period float
     [-divide_by integer] [-instance name]
     [-scan_clock scan_clock]
     [-shift_enable test_signal] [-design design]
```

Defines an `OPCG` (clock) domain.

The command returns the directory path to the `opcg_domain` object that it creates.You can find the objects created by the `define_dft opcg_domain` constraints in:

`/designs/top/dft/opcg/opcg_domains/`

Options and Arguments

`-counter_length` *`integer`*

Specifies the number of bits in the down counter register.

Specify a number between 4 and 7, or specify 0 if no counter should be inserted.

Note: This option cannot be specified with the `-domain_macro_parameter` and `-location` options.

`-design` *`design`*

Specifies the name of the design for which the domain is defined.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

This option is required if multiple designs are loaded.

`-divide_by` *`integer`*

Specifies to build an internal clock divider circuit which divides the oscillator source frequency by the specified number.

`-domain_macro_parameter` *`domain_macro_parameter`*

Specifies the domain macro parameter set to be used for the `OPCG` domain.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The domain macro parameter set must have been previously defined by the `define_dft domain_macro_parameters` command.

`-instance` *`name`*

Specifies the name of the OPCG domain instance.

Note: This option cannot be specified with the `-domain_macro_parameter`, `-location` and `-divide_by` options.

`-location {`*`subport`* `|` *`port`* `|` *`pin`*`}`

Specifies where to insert the domain macro. Specify an existing top-level port, subport, or hierarchical pin name.

`-max_num_pulses` *`integer`*

Specifies the number of high speed pulses the domain macro should be able to generate. The maximum number you can specify is 8.

Default: `2`

Note: This option cannot be specified with the `-domain_macro_parameter` and `-location` options.

`-min_domain_period` *`float`*

Specifies the minimum period (in picoseconds) at which the `OPCG` domain is assumed to operate.

Note: The frequency at which the domain is supposed to operate (specified using this option) must be greater than the minimum frequency of the oscillator source, specified using the `-max_output_period` of the `define_dft osc_source` command.

`-name` *`name`*

Specifies the `opcg_domain` object name of the `OPCG` domain.

Note: The name allows you to search for the object later (through the `find` command) or to recognize it in reports.

Default: `OPCG_DOMAIN_n`

`-opcg_trigger` *`opcg_trigger`*

Specifies the trigger signal that will enable high speed pulses to be generated by the domain macros controlled by this trigger.

The trigger signal must have been previously defined with the `define_dft opcg_trigger` constraint.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-osc_source` *`osc_source`*

Specifies the oscillator source to be used for the `OPCG` domain.

The oscillator source must have been previously defined by a define_dft osc_source constraint.

`-scan_clock` *`scan_clock`*

Specifies the test clock that will be used for shifting the OPCG domain macro logic in fullscan mode.

If you omit this option, the scan clock specified with the `insert_dft opcg` command is used.

`-shift_enable` *`test_signal`*

Specifies the shift-enable (SE) test signal to be connected to the specified OPCG domain when OPCG is inserted, giving you control over the SE signal to use for each domain.

Examples

- The following command inserts the domain macro after output pin `Y` of buffer instance `i_buf1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_buf1/Y -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro before input pin `A` of buffer instance `i_buf1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_buf1/A -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro after pin `out1` of subdesign instance `i_core`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/out1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro within the subdesign for instance `i_core` prior to output pin `out1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/subports_out/out1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro before the `in1` pin of subdesign instance `i_core`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/in1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro within the subdesign for instance `i_core` after the input pin `in1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location  i_core/subports_in/in1 -min_domain_period 1000 -divide_by 6
```

Related Information

Affected by these constraints:	define_dft domain_macro_parameters on page 927
	define_dft opcg_trigger on page 953
	define_dft osc_source on page 956
Affects this command:	insert_dft opcg on page 1063
Sets these attributes:	OPCG Domain Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft opcg_mode

```
define_dft opcg_mode [-name name]
     -mode_init file
     [-jtag_controlled]
     [-osc_source_parameters string]...
     [-design design]
```

Defines an OPCG mode for ATPG.

The command returns the directory path to the `opcg_mode` object that it creates.You can find the objects created by the `define_dft opcg_mode` constraints in:

`/designs/top/dft/opcg/opcg_modes/`

Options and Arguments

`-design` *`design`*	Specifies the name of the design for which the `OPCG` mode is defined. This option is only required if multiple designs are loaded. If there is only one top-level design, you can omit the design name. In this case, the tool will use the top-level design of the design hierarchy.
`-jtag_controlled`	Specifies whether a JTAG instruction is used to lock the PLLs for OPCG operation.
`-mode_init` *`file`*	Specifies the name of file containing the `OPCG` mode initialization sequence that ensures that PLLs are properly initialized and locked on the input oscillators.
`-name` *`name`*	Specifies the `opcg_mode` object name of the `OPCG` mode. *Default*: *`DFT_prefix`*`_opcg_mode_n`
`-osc_source_paremeters` *`string`*	Specifies the oscillator source parameters specific for this `OPCG` mode. Use the following format: `{`*`name osc_source_output_period ref_clock_period`*`}` where *`name`* is the name of the `osc_source` object.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command defines an OPCG mode whose name defaults to `DFT_opcg_mode_1`.

```
define_dft opcg_mode -osc_source_parameters {OSC1 52.5 270.3} \
  -osc_source_parameters {OSC2 52 290} -mode_init opcgMode1.mode_init
```

Related Information

Affects this command:	write_et_atpg on page 1181
Related constraint:	define_dft osc_source on page 956
Sets these attributes:	OPCG Mode Attributes
	Osc Source Reference Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft opcg_trigger

```
define_dft opcg_trigger [-name name]
     -osc_source osc_source
     [-active {low|high}]
     [{pin|port}] [-create_port]
     [-inside hierarchical_instance]
     [-scan_clock scan_clock]
     [-delay_cycles integer | -instance instance]
     [-test_signal test_signal][-design design]
```

Defines the trigger signal that will enable the trigger macro associated with the specified oscillator source.

Note: Multiple trigger macros, when inserted, can share the same trigger signal pin.

The command returns the directory path to the `opcg_trigger` object that it creates.You can find the objects created by the `define_dft opcg_trigger` constraints in:

```
/designs/top/dft/opcg/opcg_triggers/
```

Options and Arguments

`-active {low | high}`

Specifies the active value for the `OPCG` trigger signal.

`-create_port`

Specifies whether to create the port if it does not exist.

`-delay_cycles integer`

Specifies the number of extra osc_source cycles to be used to delay the trigger signal inside the trigger macro when it is shared with a test signal.

Note: This option cannot be specified with `-instance` option.

`-design design`

Specifies the name of the design for which the trigger enable is defined.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

This option is required if multiple designs are loaded.

`-inside hierarchical_instance`

Specifies the hierarchical instance in which the `OPCG` trigger must be inserted. This instance will be uniquified if needed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: inserted at the top-level of the design.

`-instance` *`instance`* Specifies the name of the OPCG trigger instance.

`-name` *`name`* Specifies the `opcg_trigger` object name of the `OPCG` trigger signal.

Default: `OPCG_TRIGGER_`*`n`*

`-osc_source` *`osc_source`*

Specifies the object name of the oscillator source whose trigger macro is enabled by the defined signal.

The oscillator source must have been defined with `define_dft osc_source` command.

`{`*`pin`*`|`*`port`*`}` Specifies the driving (input) pin or port for the `OPCG` trigger signal.

Note: If multiple designs are loaded and you did no specify the `-design` option, you can also specify the full path to the driver.

`-scan_clock` *`scan_clock`*

Specifies the test clock that must be used for shifting the `OPCG` trigger logic in fullscan mode.

If you omit this option, the scan clock specified with the `insert_dft opcg` command is used instead.

`-test_signal` *`test_signal`*

Specifies the test signal for the OPCG trigger.

Example

The following command defines the `opcg_trigger` object associated with oscillator source `OSC` on port `GO`. This port will be created if it does not exist.

```
define_dft opcg_trigger -name TRIGGER -create_port  -active low  -osc_source OSC GO
```

Related Information

Affects this command: insert_dft opcg on page 1063

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related constraints:	define_dft osc_source on page 956
	define_dft opcg_domain on page 947
	define_dft opcg_mode on page 951
Sets these attributes:	OPCG Trigger Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft osc_source

```
define_dft osc_source [-name osc_source]
     -ref_clock_pin {pin|port}
     -min_input_period integer
     -max_input_period integer
     -min_output_period integer
     -max_output_period integer
     pin
     [-design design]
```

Defines an oscillator source on the output pin of a PLL instance that will drive the `OPCG` logic.

The command returns the directory path to the `osc_source` object that it creates.You can find the objects created by the `define_dft osc_source` constraints in:

```
/designs/top/dft/opcg/osc_sources/
```

Options and Arguments

`-design` *design*

Specifies the name of the design for which the oscillator clock is defined.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

This option is required if multiple designs are loaded.

`-max_input_period` *float*

Specifies the maximum period of the reference (input) clock. Specify the clock period in picoseconds. The value of this option must be larger than the value of the `-min_input_period` option.

`-max_output_period` *float*

Specifies the maximum period of the generated (output) clock. Specify the clock period in picoseconds. The value of this option must be larger than the value of the `-min_output_period` option.

`-min_input_period` *float*

Specifies the minimum period of the reference (input) clock. Specify the clock period in picoseconds.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-min_output_period` *`float`*	Specifies the minimum period of the generated (output) clock. Specify the clock period in picoseconds.
`-name` *`osc_source`*	Specifies the name of the `osc_source` object for the oscillator source. *Default*: `OSC_SOURCE_`*`n`*
`pin`	Specifies the output pin of the PLL.
`-ref_clock_pin` *`pin`*	Specifies the reference (input) clock pin or port for the PLL.

Example

The following command defines an oscillator source for instance `instPLL1` whose reference (input) clock has a period between 20000 and 4000 ps (or clock frequency between 50 and 250 MHz) and whose output clock has a period between 2000 and 500 ps (or a clock frequency between 500 and 2000 MHz).

```
define_dft osc_source -name instPLL1
    -ref_clock_pin REFCLOCKPORT1 \
    -min_input_period 4000 -max_input_period 20000
    -min_output_period 500 -max_output_period 2000 \
    PLL1/Z1
```

Related Information

Affects these commands	define_dft opcg_domain on page 947
	define_dft opcg_mode on page 951
	define_dft opcg_trigger on page 953
	iinsert_dft opcg on page 1063
Sets these attributes:	Osc Source Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft pmbist_direct_access

```
define_dft pmbist_direct_access -function string
     -active {low | high}
     [-hookup_pin pin ]
     [-hookup_polarity {non_inverted | inverted}]
     {mbist_clock | port} [-fcu]
     [-design design]
```

Defines programmable MBIST direct access interface pins or ports which can be used as an alternative access mechanism for PMBIST. This method can be used as a supplement to or replacement for the JTAG interface for controlling MBIST.

Note: All signals are level-sensitive. The patterns that you generate by running the `create_embedded_test` command in Modus properly stimulate and monitor the defined signals if they are directly accessible from design ports.

Options and Arguments

`-active {low|high}`	Specifies the active value of the signal.
`-design` *`design`*	Specifies the name of the design for which the domain macro parameter set is defined. If you omit the design name, the top-level design of the current directory of the design hierarchy is used. This option is required if multiple designs are loaded.
`-fcu`	Specifies that the function is associated with the fuse control unit.
`-function` *`string`*	Defines the functionality of the direct access pin. The option can have one of the following values: ■ `mda_done` Specifies the pin or port where the PMBIST logic will assert the signal when memory testing is complete for direct access or the first full loop has completed for burning patterns.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `mda_fail`

  Specifies the pin or port where the PMBIST logic will assert the signal when a failure occurs during memory testing. The signal is asserted only for direct access patterns and not for burning patterns.

- `mda_reset`

  Specifies the pin or port to asynchronously reset the PMBIST logic for direct access. This can be JTAG signal for `trst` if TRST is not held active during functional mode. This function is mandatory.

- `mda_tck`

  Specifies the defined MBIST clock which is used to pass control and data into and extract results from the PMBIST logic on the direct access interface. This works in the same fashion as that `tck` signal in JTAG mode of operation. This function is mandatory.

- `mda_tdi`

  Specifies the data and control input port or pin for direct access PMBIST. This single signal works in a similar fashion as the combination of JTAG `tms` and `tdi` in JTAG mode of operation. This function is mandatory.

- `mda_tdo`

  Specifies the pin or port where the output of direct access PMBIST Test Data Registers can be monitored. This allows for more detailed failure information to be extracted from the `MBISTCHK` Test Data Register.

- `mda_ovfl`

  Specifies the pin or port where the PMBIST logic will assert the signal when failures overflow during memory testing for direct access.

- `mda_run`

  Specifies the input port or pin for on-demand PMBIST. This single signal works as a 'go' signal for on-demand PMBIST (like power-on-self-test).

- `mda_run_mask`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the input port or pin which is used to control testplans for on-demand PMBIST.

- `mda_run_done`

  Specifies the pin or port where the PMBIST logic will assert the signal when memory testing is complete for on-demand PMBIST.

- `mda_run_fail`

  Specifies the pin or port where the PMBIST logic will assert the signal when a failure occurs during memory testing. The signal is asserted only for on-demand PMBIST.

- `mda_run_ovfl`

  Specifies the pin or port where the PMBIST logic will assert the signal when failures overflow during memory testing for on-demand PMBIST.

`{`*`mbist_clock_domain`* `|`*`port`*`}`

Specifies the source port or mbist clock object for this function. For function `mda_tck`, it must be an `mbist_clock` object. For rest of the functions, it can be a unidirectional port for the block level flow and it can be any pin or a port for the chip level flow.

Any port can be shared for functional uses provided the defined and applied PMBIST DFT configuration mode enables PMBIST usage during PMBIST operations.

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the direct access pin.

Note: When you specify this option, the Genus-DFT engine does not validate the controllability of any logic between the top-level signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {inverted|non_inverted}`

Specifies the polarity of the signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	insert_dft pmbist on page 1065
Sets these attributes:	Programmable Direct Access Function Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft pmbist_port

```
define_dft pmbist_port -function string
     [-design design] string
```

Allows the user to define a user specified name for a particular function for PMBIST at block level.

Options and Arguments

`-design` *`design`*	Specifies the name of the design for which the port is defined. If you omit the design name, the top-level design of the current directory of the design hierarchy is used. This option is required if multiple designs are loaded.
`-function` *`string`*	Specifies the function of the port. These include (`tdi` | `tdo` | `reset` | `shiftdr` | `runidle` | `capturedr` | `updatedr` | `fcu_rdi` | `fcu_rdo` | `fcu_clk` | `fcu_active` | `fcu_transfer` | `temmda_tdo` | `temmda_tdi` | `retention_pause_active` | `retention_pause_continue` | `decode_<default instruction name>`)
`string`	Specifies the name of the port to be used for the specified function.

Related Information

Affected attributes: Programmable Direct Access Function Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft preserved_segment

```
define_dft preserved_segment [-name segment_name]
     { {instance|segment_name}... [-sdi pin] [-sdo pin]
     | -analyze -sdi {pin|port} -sdo {pin|port} }
     [ -connected_shift_enable
     |[-connected_scan_clock_a][-connected_scan_clock_b]]
     [-allow_reordering]
```

Defines a preserved segment. In a preserved segment, the elements of the mapped segment are already connected in the specified order, and they cannot be reordered by a physical scan reordering tool.

A preserved segment is a user-specified scan segment which can be associated with either

- A user-defined top-level chain—created using the `define_dft scan_chain` command
- A tool-created scan chain—created using the `connect_scan_chains` command

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft preserved_segment` constraints in:

`/designs/top_design/dft/scan_segments`

Options and Arguments

`-allow_reordering`
: Indicates whether the order of the elements can be changed during scan configuration and by a physical scan reordering tool.

`-analyze`
: Analyzes the connectivity of the scan segment, and returns the elements of the segment given its endpoints.

  Note: If the segment includes degenerated scan flops in its element list, you must also specify the `-connected_shift_enable` option for the command to complete successfully.

`-connected_scan_clock_a (-connected_scan_clock_b)`
: Indicates that the `scan_clock_a` (`scan_clock_b`) port of the module boundary is driven by external logic (preconnected). The external logic connected to the `scan_clock_a` (`scan_clock_b`) pin of the module will not be modified by the scan configuration engine.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This option applies only for the clocked LSSD scan style.

`-connected_shift_enable`

Indicates that the shift enable port of the module boundary is driven by external logic (preconnected) or that the shift enable signal is internally generated within the module boundary. In either case, the external logic connected to the shift enable pin of the module, or the internal logic driving the shift enable pins of the flip-flops in the module will not be modified by the scan configuration engine.

You must specify this option to successfully analyze preserved segments (such as shift-register segments) into the RC session when these segments include degenerated scan flops (or scan flops whose instances have their SI and SE pins are tied-off).

`{`*`instance`*`|`*`segment_name`*`}`

Specifies an element in the scan segment being defined. List the elements in the order they should appear in the scan chain (in shift order, that is, left-most corresponds to first bit shifted-in). An element can be a flip-flop instance, a combinational instance, or a scan segment.

Additionally, the hierarchical scan data input and output pins of the segment can be specified using the `-sdi` and `-sdo` options respectively. If the hierarchical `SDI` and `SDO` pins of the segment are both at the boundary of the same lower module, the RC-DFT engine also traces the shift-enable signal back from the scan registers in the segment to the same module boundary. It hooks up the shift-enable signal at the module boundary whenever applicable (only if you did not specify the `-connected_shift_enable` option).

`-name` *`segment_name`* Defines a name for the segment that you can use to reference in the `define_dft scan_chain` constraint.

`-sdi (-sdo)` Specifies the scan data input (scan data output) of the segment. Specify a hierarchical pin name or port.

Examples

- The following example defines a pre-existing segment in instance `u_a` using its scan data input and output pins.

```
legacy_genus:/> define_dft preserved_segment -name segmenta -analyze \
-sdi */u_a/SIa -sdo */u_a/SOa
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example defines a pre-existing segment by specifying its hierarchical scan data input and output pins, and its elements consisting of two combinational components, four sequential registers, a scan abstract segment, and a combinational component endpoint.

```
define_dft preserved_segment -name preservedSeg \
    -sdi i_core/i_anor1/B0 -sdo i_core/bufToAnchorSeg/Y \
    i_core/i_anor1 i_core/i_anor2 \
    i_core/i_flop11 i_core/i_flop22 i_core/i_flop33 i_core/i_flop44 \
    absSeg \
i_core/bufToAnchorSeg
```

Related Information

Affects this constraint:	define_dft scan_chain on page 966
Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft scan_chain

```
define_dft scan_chain [-name name]
    {-sdi sdi -sdo sdo [-create_ports] [-shared_input]
      {-shared_output [-shared_select test_signal] |
       -non_shared_output}
     [-hookup_pin_sdi pin] [-hookup_pin_sdo pin]
     [-shift_enable test_signal]
     [-head segment] [-tail segment] [-body segment]
     [-complete | -max_length integer]
     [-domain test_clock_domain [-edge {rise|fall}]]
     [-terminal_lockup {level_sensitive|edge_sensitive}]
     [-configure_pad {tm_signal | se_signal} ]
    |-analyze -sdo sdo [-sdi sdi] [-dont_overlay]
      {-shared_out  | -non_shared_out} }
```

Creates a scan chain or analyzes an existing chain with the specified input and output scan data ports.

If you created a scan chain, the command returns the directory path to the `scan_chain` object that it creates. For newly created scan chains you can find the objects created by the `define_dft scan_chain` constraints in:

`/designs/top_design/dft/scan_chains`

If you successfully analyzed an existing scan chain, the command returns the directory path to the `actual_scan_chain` object that it creates. For successfully analyzed scan chains, you can find the objects created by the `define_dft scan_chain` constraints in:

`/designs/top_design/dft/report/actual_scan_chains`

Options and Arguments

`-analyze`	Analyzes the connectivity of an existing top-level scan chain in a structural netlist compiled in a previous RC session. You must at least specify the scan data output pin and optionally the scan data input pin to identify the chain.
`-body segment`	Indicates that the specified segment (an ordered set of scan flip-flops) is part of the body of the scan chain. The segment must have been previously defined with a `define_dft xxx_segment` constraint, where xxx is either `abstract`, `fixed`, `floating`, `preserved`, or `shift_register`.
`-complete`	Specifies that the defined chain is complete and no other flip-flops should be added to the chain.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal (test mode or shift enable signal) that the Genus-DFT engine must use if it needs to configure the pad connected to the scan data input or output signal to control the data direction during test mode.

Tip

If the scan data input and output pin are shared with functional pins, you should use the shift enable test signal to configure pads. This will allow the pads to shift-in and shift-out data when shift-enable signal is active (scan-shift mode), and will allow the pads to operate in functional mode when shift-enable signal is inactive (capture mode).

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-create_ports`

Specifies whether to create the scan data input and output ports if they do not exist.

If you do not specify the scan data input or output signals using the `-sdi` and `-sdo` options, the ports can be created and named as *`prefix`*`_sdi_`*`num`* and *`prefix`*`_sdo_`*`num`*, where *`prefix`* is the value of the `dft_prefix` attribute.

`-domain test_clock_domain`

Specifies the DFT clock domain to associate with the scan chain. This clock domain must have been previously identified by the `check_dft_rules` command or defined with the `define_dft test_clock` constraint.

If you omit this option, and segments have been defined for the chain, the scan chain is automatically associated with the appropriate DFT clock domain. In the absence of segments, the scan chain can be assigned to any DFT clock domain.

Note: This option only applies to the muxed scan style.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dont_overlay`

Prevents that Genus re-associates (or overlays) user-defined segments of type preserve or fixed to its analyzed scan chains. Consequently, the segments are not re-established as a fixed entity (and hence are reorderable) when determining how to partition the chains for physical-based reordering. If these segments include multi-input combinational logic gates in the scan data path, the `write_scandef` command uses these gates to partition the analyzed scan chains into n-reorderable segments (referred to as scanDEF chains). The register before a multi-input combinational logic gate becomes the `STOP` point for one scanDEF chain, while the register after a combinational gate becomes the `START` point of another scanDEF chain.

`-edge {rise|fall}`

Specifies whether to use the falling or rising edge of the test clocks in the specified DFT clock domain. You can specify this option only if you specified `-domain`.

If you omit this option, the scan flip-flops triggered by the different active edges of the test clocks will be placed on their own scan chain.

This option is ignored if you enabled the `dft_mix_clock_edges_in_scan_chains` attribute.

Note: This option only applies to the muxed scan style.

`-head` *`segment`*

Indicates that the specified segment (an ordered set of scan flip-flops) must be placed at the head (closest to the scan data input) of the scan chain. The segment must have been previously defined with a `define_dft xxx_segment` constraint, where xxx is either `abstract`, `fixed`, `floating`, `preserved`, or `shift_register`.

`-hookup_pin_sdi` *`pin`*

Specifies the core-side hookup pin to be used for the scan data input signal during scan chain connection.

Note: When you specify this option, the Genus-DFT engine does not validate the control ability of any logic between the top-level scan data input signal and its designated hookup pin under test-mode setup.

`-hookup_pin_sdo` *`pin`*

Specifies the core-side hookup pin to be used for the scan data output signal during scan chain connection.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: When you specify this option, the Genus-DFT engine does not validate the control ability of any logic between the top-level shift_enable signal and its designated hookup pin under test-mode setup.

`-max_length` *`integer`*

Specifies the maximum length that you allow for this scan chain.

If you omit this option, the maximum length defaults to the value of the `dft_max_length_of_scan_chains` design attribute.

Note: This option is ignored if the scan chain is defined with the `-complete` option, or if the number of flip-flop instances in a head, body, or tail segment exceeds the maximum value.

`-name` *`name`*

Specifies the name of the scan chain.

If you omit this option, a default name is used.

`-sdi` *`sdi`*

Specifies the scan data input signal.

- If you want to *create* a chain, specify a top-level port or a hierarchical pin name in case of an existing port or pin. If you want the tool to create the port, use the `-create_ports` option and a primary input port with the specified name will be created.
- If you want to *analyze* an existing chain, specify a top-level port, a hierarchical pin, subport, or a non-sequential instance pin.

`-sdo` *`sdo`*

Specifies the scan data output signal.

- If you want to *create* a chain, specify a top-level port or a hierarchical pin name in case of an existing port or pin. If you want the tool to create the port, use the `-create_ports` option and a primary output port with the specified name will be created.
- If you want to analyze an existing chain, specify a top-level port, a hierarchical pin, subport, or a non-sequential instance pin.

`-shared_input`

Specifies whether an existing functional input port can be used as scan data input port.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-shared_select` *`test_signal`*

Specifies the select control signal to the mux inserted for a shared output port.

Default: The default shift-enable signal or chain-specific shift-enable signal is used as the select control signal to the mux.

{`-shared_output` | `-non_shared_output`}

Specifies whether an existing functional output port can be used as scan data output port. If the functional port can be used for scan data purposes, a mux is inserted in the scan data path by the `connect_scan_chains` command.

One of these options must be specified when the specified scan data output signal is already connected in the circuit.

`-shift_enable` *`test_signal`*

Designates a chain-specific shift-enable port or pin.

If you omit this option, the default shift-enable signal specified using a `define_dft shift_enable` constraint is used.

`-tail` *`segment`*

Indicates that the specified segment (an ordered set of scan flip-flops) must be placed at the tail (closest to the scan data output) of the scan chain. The segment must have been previously defined with a `define_dft xxx_segment` constraint, where xxx is either `abstract`, `fixed`, `floating`, `preserved`, or `shift_register`.

`-terminal_lockup {level_sensitive | edge_sensitive}`

Specifies the type of lockup element that configuration can insert at the tail end of the chain to connect to the specified scan data output signal.

If this option is not specified, no terminal lockup element will be inserted.

Note: This option only applies to the muxed scan style.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example creates a chain containing 3 segments previously defined:

```
legacy_genus:/des*/test> define_dft scan_chain -sdi in[0] -sdo out[0] \
 -shared_out -head segHead -tail segTail -body segBody -name chain1

...
Info    : Added scan chain. [DFT-151]
        : scan chain successfully defined.
/designs/test/dft/scan_chains/chain1
```

- The following example analyzes an existing scan chain:

```
legacy_genus:/> define_dft scan_chain -name topChain -sdi SI -sdo SO -analyze

...
Info    : Added scan chain. [DFT-151]
        : scan chain successfully defined.
/designs/test/dft/report/actual_scan_chains/topChain
```

Related Information

Affected by this constraint:	define_dft abstract_segment on page 915
	define_dft fixed_segment on page 929
	define_dft floating_segment on page 931
	define_dft preserved_segment on page 963
	define_dft shift_enable on page 978
	define_dft shift_register_segment on page 981
	define_dft test_clock on page 990
Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
Affected by this attribute:	dft_max_length_of_scan_chains
	dft_mix_clock_edges_in_scan_chains
	dft_prefix
Sets these attributes:	Scan Chain Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft scan_clock_a

```
define_dft scan_clock_a
     [-name name] [-ideal] driver
     [-period integer] [-divide_period integer]
     [-rise integer] [-divide_rise integer]
     [-fall integer] [-divide_fall integer]
     [ [-hookup_pin pin [-hookup_polarity string]]
       [-configure_pad {tm_signal|se_signal}]
     | [-create_port]]
```

Defines the scan clock of the master latch (`scan_clock_a`) of the clocked LSSD scan cell. The `scan_clock_a` signal controls the scan shifting of the master latch and is required for the clocked-LSSD scan style. The signal is created with active high polarity.

You can define only one `scan_clock_a` signal for the design. If you specify more than one signal, the last definition overwrites the existing one.

The command returns the virtual directory path to the `scan_clock_a` signal object that it creates. You can find the object created by the `define_dft scan_clock_a` constraints in:

`/designs/design/dft/test_signals`

Options and Arguments

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the `scan_clock_a` signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-create_port`

Specifies whether to create the port if it does not exist.

`-divide_fall integer`

Together with the `-fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `100`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-divide_period` *`integer`*

Together with the `-period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

Default: `1`

`-divide_rise` *`integer`*

Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`.

Default: `100`

`driver`

Specifies the driving pin or port for the scan clock of the master latch (`scan_clock_a`) of the clocked LSSD scan cell.

`-fall` *`integer`*

Together with the `-divide_fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `60`

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the `scan_clock_a` signal during scan chain connection.

Note: When you specify this option, the Genus-DFT engine does not validate the controllability of any logic between the top-level `scan_clock_a` signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {inverted|non_inverted}`

Specifies the polarity of the `scan_clock_a` signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-ideal`

Marks the `scan_clock_a` signal as ideal. This prevents buffering of the `scan_clock_a` network during optimization.

By default, the scan_`clock_a` signal is marked non-ideal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Note: If the test signal is marked as ideal, Genus sets the `ideal_network` attribute to `true` on the pin or port for the `scan_clock_a` signal
`-name` *`name`*	Specifies the `test_signal` object name of the `scan_clock_a` signal. If you omit this option, the Genus-DFT engine assigns a name based on the hierarchical path of the driver, using underscores as delimiters in the path.
`-period` *`integer`*	Together with the `-divide_period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`. *Default*: `50000` (20 MHz test clock)
`-rise` *`integer`*	Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`. *Default*: `50`

Example

- The following example defines `sca` as the driver for the `scan_clock_a` signal and assigns `SCA` as the `test_signal` object name.

```
define_dft scan_clock_a -name SCA sca
```

Related Information

Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
	write_atpg on page 1148
Sets these attributes:	Test Signal Attributes
Affected by this attribute:	dft_scan_style

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft scan_clock_b

```
define_dft scan_clock_b
     [-name name] [-ideal] driver
     [-period integer] [-divide_period integer]
     [-rise integer] [-divide_rise integer]
     [-fall integer] [-divide_fall integer]
     [ [-hookup_pin pin [-hookup_polarity string]]
       [-configure_pad {tm_signal|se_signal}]
     | [-create_port]]
```

Defines the scan clock of the slave latch (`scan_clock_b`) of the clocked LSSD scan cell. The `scan_clock_b` signal controls the scan shifting of the slave latch and is required for the clocked-LSSD scan style. The signal is created with active high polarity.

You can define only one `scan_clock_b` signal for the design. If you specify more than one signal, the last definition overwrites the existing one.

The command returns the virtual directory path to the `scan_clock_b` object that it creates. You can find the object created by the `define_dft scan_clock_b` constraints in:

`/designs/design/dft/test_signals`

Options and Arguments

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the `scan_clock_b` signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-create_port`

Specifies whether to create the port if it does not exist.

`-divide_fall integer`

Together with the `-fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `100`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-divide_period` *`integer`*

Together with the `-period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

Default: `1`

`-divide_rise` *`integer`*

Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`.

Default: `100`

`driver`

Specifies the driving pin or port for the scan clock of the slave latch (`scan_clock_b`) of the clocked LSSD scan cell.

`-fall` *`integer`*

Together with the `-divide_fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `80`

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the `scan_clock_b` signal during scan chain connection.

Note: When you specify this option, the Genus-DFT engine does not validate the controlability of any logic between the top-level `scan_clock_b` signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {inverted|non_inverted}`

Specifies the polarity of the `scan_clock_b` signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-ideal`

Marks the `scan_clock_b` signal as ideal. This prevents buffering of the `scan_clock_b` network during optimization.

By default, the `scan_clock_b` signal is marked non-ideal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Note: If the test signal is marked as ideal, Genus sets the `ideal_network` attribute to `true` on the pin or port for the `scan_clock_b` signal
`-name` *`name`*	Specifies the `test_signal` object name of the `scan_clock_b` signal. If you omit this option, the Genus-DFT engine assigns a name based on the hierarchical path of the driver, using underscores as delimiters in the path.
`-period` *`integer`*	Together with the `-divide_period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`. *Default*: `50000` (20 MHz test clock)
`-rise` *`integer`*	Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`. *Default*: `70`

Example

- The following example defines `sca` as the driver for the `scan_clock_b` signal and assigns `SCB` as the `test_signal` object name.

```
define_dft scan_clock_b -name SCB scb
```

Related Information

Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
	write_atpg on page 1148
Sets these attributes:	Test Signal Attributes
Affected by this attribute:	dft_scan_style

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft shift_enable

```
define_dft shift_enable [-name name] -active {low|high}
     [-default] [-ideal]
     [ [-hookup_pin pin [-hookup_polarity string]]
       [-configure_pad {tm_signal|se_signal}]
     | -create_port ]
     [-lec_value {auto | 0 | 1 | no_value }]
     {pin|port} [-design design]
```

Specifies the name and active value of the input signal that activates scan shifting. The input signal can be defined on a top-level port or an internal driving pin. This type of input signal is required by the `muxed_scan` style. The active value of the shift-enable signals is propagated through the design by the `check_dft_rules` command.

The command returns the directory path to the `test_signal` object that it creates. You can find the objects created by the `define_dft shift_enable` constraints in:

`/designs/design/dft/test_signals`

Options and Arguments

`-active {low | high}`

Specifies the active value for the shift-enable signal.

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the shift-enable signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft test_mode` or `define_dft shift_enable` constraint.

`-create_port`

Specifies whether to create the port if it does not exist.

`-default`

Designates the specified signal as the default shift-enable signal for chains for which you omit the `-shift-enable` option.

Note: You can designate only one signal as the default shift-enable signal.

Note: If no shift-enable is defined as the default enable signal, the first-defined shift-enable signal is used as the default enable.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-design` *`design`*

Specifies the name of the design for which the shift enable is defined.

If you omit the design name and multiple designs are loaded, the top-level design of the current directory of the design hierarchy is used.

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the top-level shift-enable signal during DFT synthesis.

Note: When you specify this option, the Genus-DFT engine does not validate the controlability of any logic between the top-level shift_enable signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {non_inverted | inverted}`

Specifies the polarity of the shift-enable signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-ideal`

Marks the shift-enable signal as ideal. This prevents buffering of the shift-enable network during optimization.

Default: The shift-enable signal is marked non-ideal.

Note: If the test signal is marked as ideal, Genus sets the `ideal_network` attribute to `true` on the pin or port for the shift-enable signal

`-name` *`name`*

Specifies the `test_signal` object name of the shift-enable signal.

If you omit this option, the Genus-DFT engine assigns a name based on the hierarchical path of the driver, using underscores as delimiters in the path.

`-lec_value {auto | 0 | 1 | no_value}`

Specifies the approach to constraining the pin for LEC validation by the `write_do_lec` command. You can specify any of the following values:

- `auto`—Writes the opposite of the test mode active value as the “add pin constraint” in the do file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- 0—Writes a logic `0` for the `add pin constraint` in the do file
- 1—Writes a logic `1` for the `add pin constraint` in the do file
- no_value—Does not write the signal as an “add pin constraint” in the do file

Default: `auto`

`{pin|port}` Specifies the driving pin or port for the shift-enable signal.

Note: If multiple designs are loaded and you did no specify the `-design` option, you can also specify the full path to the driver.

Example

- When the following constraint is given, the `check_dft_rules` command propagates a logic1 from the `p_top/SE` pin into the design.

```
define_dft shift_enable -active low -hookup_pin p_top/SE -hookup_polarity inverted
```

Related Information

Affects these commands:	check_dft_rules on page 870
	connect_scan_chains on page 906
	insert_dft shadow_logic on page 1077
	report dft_chains on page 1125
Sets these attributes:	Test Signal Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft shift_register_segment

```
define_dft shift_register_segment [-name segment_name]
     -start_flop instance
     -end_flop instance
```

Defines a shift register. Because a shift register is a shiftable scan chain segment, the Genus-DFT engine can use the functional path of the shift register as the scan path by only scan-replacing the first flop in the shift register segment, while maintaining the existing connectivity of the flops.

Note: A shift register segment can only contain flops driven by the same clock and same clock edge.

A shift register is a user-specified scan segment which can be associated with either

- A user-defined top-level chain—created using the `define_dft scan_chain` command
- A tool-created scan chain—created using the `connect scan chains` command

The command returns the virtual directory path to the created shift register segment. You can find the objects created by the `define_dft shift_register_segment` constraint in:

`/designs/top_design/dft/scan_segments`

Note: Shift register segments are only supported for the muxed scan style.

Options and Arguments

`-end_flop instance`	Specifies the last flop in the shift register. Specify the hierarchical instance name of the flop.
`-name segment_name`	Defines a name for the segment that you can use to reference in the `define_dft scan_chain` constraint.
`-start_flop instance`	Specifies the first flop in the shift register. Specify the hierarchical instance name of the flop.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example defines a shift register.

```
define_dft shift_register_segment -name myreg \
-start_flop *seq/out_reg_0 -end_flop *seq/out_reg_7
```

Related Information

Affects this constraint:	define_dft scan_chain on page 966
Affects these commands:	connect_scan_chains on page 906
	report dft_chains on page 1125
Related command:	identify_shift_register_scan_segments on page 1019
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft tap_port

```
define_dft tap_port {pin|port} [-create_port]
     -type {tck | tdi | tdo | tdo_enable | tms | trst}
     [-hookup_pin {pin|port}] [-hookup_polarity string]
     [-tck_period integer] [-no_trst_test_signal]
     [design]
```

Defines a TAP port (JTAG signal). Specifies the internal hookup pin for a JTAG signal. These hookup pins will be used when inserting boundary scan logic, JTAG macro, MBIST logic and PTAM logic, to make connections from its DFT logic to the hookup pins specified for each of the JTAG signals.

Note: You can only specify this command before inserting the boundary scan logic or the JTAG macro. When defining an existing JTAG macro using `define_dft jtag_macro`, the tap ports must be defined with the `define_dft tap_port` command.

You can find the objects created by the `define_dft tap_port` constraints in:

`/designs/design/dft/boundary_scan/tap_ports`

Options and Arguments

`-create_port`	Creates the port if it does not exist.
design	Specifies the design for which the tap ports are being defined.
`-hookup_pin {pin | port}`	Specifies the core-side hookup pin to be used for the top-level JTAG signal during DFT synthesis.
`-hookup_polarity {non_inverted | inverted}`	Specifies the polarity of the JTAG signal at the core-side hookup pin. By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.
`-no_trst_test_signal`	Prevents defining an active-low test signal on the TRST port and causes a formal verification constraint with a value of ‘0’ to be added for the TRST port for the revised design.
`{pin| port}`	Specifies the driving pin or port for the JTAG signal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-tck_period` *`integer`*

Specifies the frequency (in picoseconds) of the Test Clock (TCK port) on the TAP Controller.

Note: You can only specify this option when you have set the `-type` option to `tck`.

Default: `50000`

`-type {tck | tdi | tdo | tdo_enable | tms | trst}`

Specifies the type of TAP port being created.

Related Information

Sets these attributes: TAP Port Attributes

dft_tap_tck_period

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft test_bus_port

```
define_dft test_bus_port
     [-name string] -function string [-index integer]
     [-hookup_pin {pin|port}] [-hookup_polarity string]
     [-wir_signal  [-wir_reset_value {low|high}]
       [-wir_tm_value {low|high}] ]
     [-create_port][-lib_cell base_cell]{pin|port}
```

Defines a test bus port on any port.

Use the `-function` option to specify the functionality of the port in test mode, such as scan data input, scan data output, scan control pin, and so on. For multiple test bus ports with the same function, use the `-index` option to distinguish them.

You can find the objects created by the `define_dft test_bus_port` constraints in:

`/designs/design/dft/test_bus_ports`

Options and Arguments

`-create_port` Creates the port if it does not exist.

`-function string` Specifies the function of the test bus port. Following values can be specified

Function	Purpose
`async_set_reset`	Test-mode signal that controls the asynchronous set/reset pin of a flop
`aux_scan_clock`	Reserved for future use
`broadcast_enable`	Broadcast versus Daisy chain mode control signal
`capture_wr`	`WIR` capture control
`codec_pipe_bypass`	Reserved for future use
`comp_pipe_bypass`	Reserved for future use
`compress_sdi`	Common scan data input (broadcast to multiple cores)
`compress_sdo`	Common scan data output

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`compression_clock`	Test clock used for compression mask (and MISR if present)
`compression_enable`	Control to enable compression mode
`cpi_over_npi`	Reserved for future use
`custom`	User specified control signal
`decomp_pipe_bypass`	Reserved for future use
`elastic_reset`	Reserved for future use
`first_core`	Reserved for future use
`in_pipe_bypass`	Reserved for future use
`lbist_direct_access`	Reserved for future use
`lbist_test_enable`	Reserved for future use
`lssd_scan_clock_a`	Reserved for future use
`lssd_scan_clock_b`	Reserved for future use
`mask_enable`	Control to enable channel masking
`mask_load`	Control to enable mask loading mode
`mbist_retention_test_out`	Output test signal defined with `pmbist_use` attribute indicating `retention_pause_active`
`misr_bypass`	Reserved for future use
`misr_reset_enable`	Control signal to reset the MISR (misr compression only)
`opcg_enable`	Control to enable OPCG mode
`opcg_load`	Control to enable concatenation of the OPCG side-scan registers across cores
`opcg_load_clock`	OPCG side scan load clock
`opcg_trigger`	Control to enable Trigger Macro
`out_pipe_bypass`	Reserved for future use
`parallel_sdi`	Reserved for future use

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`parallel_sdo`	Reserved for future use
`pipe_sen`	Control pin that enables the pipeline flops. It is usually a shift enable pin.
`scan_clock`	Reserved for future use
`select_bypass`	Select 1500 bypass mode
`select_serial`	Select 1500 serial mode
`select_wir`	Select Wrapper Instruction Register (WIR) access control signal
`serial_sdi`	1500 Wrapper Serial Input
`serial_sdo`	1500 Wrapper Serial Output
`serial_wrck`	Clock used for inserted 1500 wrapper instruction registers (WIR) and 1500 bypass registers
`shift_enable`	Test signal that controls the shifting of data in a scan chain when asserted to its active value.
`shift_wr`	`WIR` shift control
`smartscan_broadcast`	Reserved for future use
`smartscan_enable`	Reserved for future use
`spread_enable`	Control to enable the xor decompressor
`test_clock`	Reserved for future use
`test_mode`	Test signal that controls the design in test mode
`update_wr`	`WIR` update control
`wext`	Wrapper Extest control signal
`wint`	Wrapper Intest control signal
`wir_test`	Test mode signal to put the WIR in its ATPG-test mode
`wrapper_and_compression_clock`	Reserved for future use

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`wrapper_clock`	Clock used for 1500 dedicated wrapper cells
`wrapper_reset`	Wrapper instruction register reset control

Note: Some restrictions apply for the functions. Refer to *Genus Design for Test Guide for Legacy UI* for more information.

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the top-level signal during DFT synthesis.

Note: When you specify this option, the Genus-DFT engine does not validate the controllability of any logic between the top-level signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {non_inverted | inverted}`

Specifies the polarity of the signal at the core-side hookup pin.

By default, a non inverted polarity is assumed. The Genus-DFT engine does not validate the polarity at the specified hookup pin.

`-index` *`integer`*

Specifies the index of the test bus port.

When more than one test bus port exists with the same function, they will have different indexes. Examples are:

- In the case of an N-bit scan data input bus, each port has the `compress_sdi` function, but the index varies between `0` and `N-1`.
- In the case of `wide2` masking, two `mask_enable` test bus ports are defined, the first with an index of 0, the second with an index of 1.

Note: You cannot specify the `-index` option when the `-wir_signal` option is specified.

`-lib_cell` *`base_cell`*

Specifies the *`base_cell`* where the test bus port is defined.

`-name` *`string`*

Specifies the name of the test bus port.

Default: *`dft_prefix_port`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

where *dft_prefix* is the value of the dft_prefix root attribute and port is the name of the port on which the test bus port is defined.

{*pin*|*port*} Specifies the driver for the test bus port.

-wir_reset_value {low|high}

Specifies the value that the test bus port will be driven to when the Wrapper Instruction Register (WIR) is reset.

You can only specify this option if you specified the -wir_signal option.

Default: high for test bus ports with the select_serial or select_bypass function, and low for all other functions.

-wir_signal Specifies that the test bus port will be driven by a Wrapper Instruction Register (WIR) output. When you invoke the insert_dft wir_signal_bits command, a bit is inserted in the WIR register for this test bus port.

Note: You can only specify this option for the following function types:compression_enable, select_bypass, select_serial, spread_enable, wext, wint, and custom.

-wir_tm_value {low|high}

Specifies the value that the test bus port will be forced to when the test bus port with wir_test function (if one exists) is driven high and the Wrapper Instruction Register (WIR) is ATPG-tested.

You can only specify this option if you specified the -wir_signal option.

Default: the value of the -wir_reset_value option

Related Information

Affects this command:	insert_test_compression on page 1106
Sets these attributes:	Test Bus Port Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft test_clock

```
define_dft test_clock -name test_clock
     [-design design] [-domain test_clock_domain]
     [-period integer] [-divide_period integer]
     [-rise integer] [-divide_rise integer]
     [-fall integer] [-divide_fall integer]
     |-hookup_pin pin [-hookup_polarity string]]
     [-cfg_pad {tm_signal|se_signal}] [-create_port string]
     [-controllable] {pin|port} [{pin|port}]...[-function
     {test_clock|dft_clock|compressor_clock}]
```

Defines a test clock and associates a test (scan shift) clock waveform with the clock. The test clock waveform can be different from the system clocks.

If you do not define test clocks, the DFT rule checker automatically analyzes the test clocks and creates these objects with a default waveform. The waveform information is useful in determining how to order scan flip-flops in a chain, and where to insert data-lockup elements in the chain.

Test clock waveforms are used to order flip-flops that belong to the same DFT test clock domain to minimize the addition of lockup elements. Flip-flops that are triggered first are ordered and connected last in a chain.

By default this test clock will be exported as a -ES signal in Modus Test Solution pinassign file generated from `write_et_atpg` command.

The command returns the directory path to the `test_clock` object that it creates.You can find the objects created by the `define_dft test_clock` constraints in:

`/designs/design/dft/test_clock_domains`

Options and Arguments

`-cfg_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the test clock signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-controllable`

When specifying an internal pin for a test clock, this option indicates that the internal clock pin is controllable in test mode (for example, Built-in-Self-Test (BIST)). If you do not specify this option, the rule checker must be able to trace back from the internal pin to a controllable top-level clock pin.

Note: If you specify an internal pin as being controllable, you need to ensure that this pin can be controlled for the duration of the test cycle. The tool will *not* validate your assumption.

`-create_port` *`string`*

Specifies to create a port if it does not already exist.

`-design` *`design`*

Specifies the name of the design for which the test clock is defined.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-divide_fall` *`integer`*

Together with the `-fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `100`

`-divide_period` *`integer`*

Together with the `-period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

Default: `1`

`-divide_rise` *`integer`*

Together with the `-rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`.

Default: `100`

`-domain` *`test_clock_domain`*

Specifies the DFT clock domain associated with the test clock.

Clocks belonging to the same domain can be mixed in a chain.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you omit this option, a new DFT clock domain is created and associated with the test clock.

Flip-flops belonging to different test clocks in the same domain can be mixed in a chain. Lockup elements can be added between the flip-flops belonging to different test clocks.

`-fall` *`integer`*

Together with the `-divide_fall` option, determines the time that the falling edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-fall` by `-divide_fall`.

Default: `90`

`-function {test_clock|dft_clock|compressor_clock}`

Specifies the type of the clock.

Default: `test_clock`

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the top-level test clock during DFT synthesis. This pin will be used for all connections to or from the test signal.

Note: When you specify this option, the Genus-DFT engine does not validate the controlability of any logic between the top-level hookup pin under test-mode setup.

`-hookup_polarity {inverted | non_inverted}`

Specifies the polarity of the test clock signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-name` *`test_clock`*

Specifies the name of the test clock that is being defined.

Each clock object in your design must have a unique name. If you define a new test clock with the same name as an existing clock, an error message will be issued.

Note: The clock name allows you to search for the clock later (through the `find` command) or to recognize it in reports.

`-period` *`integer`*

Together with the `-divide_period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Default: `50000` (20 MHz test clock)
`{pin|port}`	Specifies the test clock input pin or port. If you specify multiple pins, these pins are assumed to have zero skew: they can be mixed without lockup latches.
`-rise integer`	Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`. *Default*: `50`

Example

The following example defines three test clocks, four test clock ports and two DFT clock domains.

```
define_dft test_clock -name CLK1X -domain domain_1 -period 20000 CLK1
define_dft test_clock -name CLK2X -domain domain_1 -period 20000 CLK2 CLK2b
define_dft test_clock -name CLK3X -domain domain_2 -period 20000 CLK3
```

The four test clock ports are CLK1, CLK2, CLK2b, and CLK3. Test clock CLK2X comprises equivalent test clocks CLK2 and CLK2B (they can be mixed in the same scan chain without any lockup element). Test clocks CLK1X, CLK2X belong to the same DFT clock domain and are compatible (requires lockup elements between compatible chain segments triggered by the different test clocks). Test clock CLK3X belongs to its own domain.

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*:

- Defining Test Clock Waveforms
- Defining Internal Clock Branches as Separate Test Clocks
- Defining Equivalent Test Clocks for Different Top-level Clock Pins
- Defining an Internal Pin as Test Clock

Affects these commands:

check_dft_rules on page 870
connect_scan_chains on page 906
fix_dft_violations on page 1010

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	report dft chains on page 1125
Sets these attributes:	Test Clock Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_dft test_mode

```
define_dft test_mode [-name name] -active {low | high}
     [-ideal] [-scan_shift]
     [ [-hookup_pin pin [-hookup_polarity string]]
       [-configure_pad {tm_signal|se_signal}]
     | [-create_port | -shared_input | -test_only] ]
     [-lec_value {auto | 0 | 1 | no_value }]
     {pin|port} [-multi_mode] [-design design]
```

Specifies the input signal and constant value that is assigned during a test session. The input signal can be defined on a top-level port or an internal driving pin.

The active value of the test mode signals is propagated through the design by the `check_dft_rules` command. Unless defined with the `-scan_shift` option, the test signal is expected to stay active throughout a test session.

The command returns the directory path to the `test_signal` object that it creates. You can find the objects created by the `define_dft test_mode` constraints in:

`/designs/design/dft/test_signals`

Options and Arguments

`-active {low | high}`

Specifies the active value for the test mode signal.

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the test mode signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-create_port`

Specifies whether to create the port if it does not exist.

Note: This option cannot be specified with the `-shared_in` option.

`-design design`

Specifies the name of the design for which the test mode signal is defined.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: If you omit the design name and multiple designs are loaded, the top-level design of the current directory of the design hierarchy is used. You can also specify the full path to the driver.

`-hookup_pin` *`pin`*

Specifies the core-side hookup pin to be used for the top-level test-mode signal during DFT synthesis.

Note: When you specify this option, the Genus-DFT engine does not validate the controllability of any logic between the top-level test-mode signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {inverted|non_inverted}`

Specifies the polarity of the test-mode signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-ideal`

Marks the test-mode signal as ideal. This prevents buffering of the test-mode network during optimization.

Default: The test-mode signal is marked non-ideal.

Note: If the test signal is marked as ideal, Genus sets the `ideal_network` attribute to `true` on the pin or port for the test signal.

`-lec_value {auto | 0 | 1 | no_value}`

Specifies the approach to constraining the pin for LEC validation by the `write_do_lec` command. You can specify any of the following values:

- `auto`—Writes the opposite of the test mode active value as the “add pin constraint” in the do file
- 0—Writes a logic `0` for the `add pin constraint` in the do file
- 1—Writes a logic `1` for the `add pin constraint` in the do file
- no_value—Does not write the signal as an “add pin constraint” in the do file

Default: `auto`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-multi_mode`

Indicates that the test signal is used for multi-mode configuration purposes.

`-name` *`name`*

Specifies the `test_signal` object name of the test signal.

If you omit this option, the Genus-DFT engine assigns a name based on the hierarchical path of the driver, using underscores as delimiters in the path.

`{`*`pin`*`|`*`port`*`}`

Specifies the driving pin or port for the test signal.

Note: If multiple designs are loaded and you did no specify the `-design` option, you can also specify the full path to the driver.

`-scan_shift`

Indicates that this test signal should only be held to its test-mode active value during the scan shift operation of the tester cycle. This option is used to designate those test signals which must be held to their non-controlling functional values to prevent the state of the flip-flops from being asynchronous set or reset while ATPG data is being shifted into or out of the scan chains.

As a consequence of specifying this option, the test signal will be treated as a non-scan clock signal by the ATPG tool. This means ATPG can pulse the pin during the capture window, but will leave it in the off state during scan shifting.

If this option is not specified, the test signal will be held to its test-mode active value for the duration of the tester cycle.

Important

Specify this option for the appropriate test signals (such as asynchronous set and reset signals and similar signals) to ensure that these test signals will not get constrained in the `write_do_lec` dofile. Not specifying this option for the appropriate test signals will result in over constraining the `write_do_lec` dofile which can lead to false EQs.

`-shared_input`

Specifies whether the input port is also used as a functional port.

By default, the signal applied to the specified driving pin or port is considered to be a dedicated test signal.

Note: This option cannot be specified with the `-create_port` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Important

Specify this option for the shared test signals (such as those driving functional logic in the golden design) to ensure that these test signals will not get constrained in the `write_do_lec` dofile. Not specifying this option for a shared test signal will result in over constraining the `write_do_lec` dofile which can lead to false EQs.

Additionally, shared test signals should not be constrained in the tool-generated DFT SDC file for the NON DFT MODE mode. By omitting this option for a shared test signal, the test signal will be listed as

- a `set_false_path` constraint when defined as an async set/reset test signal
- a `set_case_analysis` constraint when defined as a shift-enable or test-mode signal.

Either of these constraints will result in not optimizing the signal paths when the constraints are merged with the FUNCTIONAL mode.

`-test_only` Specifies that the input port will only be used for test purposes.

When this option is specified, the tool might add a constraint for the port in the do file generated by the `write_do_lec` command.

Example

- When the following constraint is given, the `check_dft_rules` command propagates a logic1 from the `pad_top/TM` pin into the design.

```
define_dft test_mode -active high -hookup_pin pad_top/TM
-hookup_polarity non_inverted test_en
```

Related Information

Affects these commands:

check_dft_rules on page 870

fix_dft_violations on page 1010

insert_dft shadow_logic on page 1077

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	insert_dft test_point on page 1083
Sets these attributes:	Test Signal Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_test_signal

```
define_test_signal {pin|hpin|port}
     -function string [-index integer] [-name string]
     [-active {high|low}] [-ideal]
     [ [-hookup_pin {pin|hpin} [-hookup_polarity string]]
       [-configure_pad {tm_signal|se_signal}]
     | [-create_port | -shared_in | -test_only] ]
     [-default_shift_enable]
     [-wir_signal  [-wir_reset_value {low|high}]
       [-wir_tm_value {low|high}] ]
     [-pipeline_depth integer]
     [-port_bus {port|port_bus|hport_bus}]
     [-lec_value {auto | 0 | 1 | no_value }]
     [-multi_mode] [-design design]
```

Defines a test bus port signal to be used in the Hierarchical Test flow.

Use the `-function` option to specify the functionality of the port in test mode, such as scan data input, scan data output, scan control pin, and so on. For multiple test bus ports with the same function, use the `-index` option to distinguish them.

You can find the objects created by the `define_test_signal` constraints in:

```
/designs/design/dft/test_signals
```

Options and Arguments

`-active {low | high}`

Specifies the active value for the test bus port signal.

Note: Only test signals with function equal to `test_mode`, `shift_enable`, `async_set_reset`, `custom`, or `wrapper_reset` can have a `low` active value.

`-configure_pad {tm_signal | se_signal}`

Specifies the test signal that the Genus-DFT engine must use if it needs to configure the pad connected to the test mode signal to control the data direction during test mode.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-create_port`

Creates the port if it does not exist.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-default_shift_enable`

Designates the specified signal as the default shift-enable signal for chains for which you omit the `-shift-enable` option.

Note: You can designate only one signal of function type `shift-enable` as the default shift-enable signal.

`-design` *`design`*

Specifies the name of the design for which the test signal is defined.

`-function` *`string`*

Specifies the function of the test bus port. Following values can be specified

Function	Purpose
`async_hold`	Reserved for future use.
`async_set_reset`	Test-mode signal that controls the asynchronous set/reset pin of a flop
`aux_scan_clock`	Reserved for future use
`broadcast_enable`	Broadcast versus Daisy chain mode control signal
`capture_wr`	`WIR` capture control
`codec_pipe_bypass`	Reserved for future use
`comp_pipe_bypass`	Reserved for future use
`compress_sdi`	Common scan data input (broadcast to multiple cores)
`compress_sdo`	Common scan data output
`compression_clock`	Test clock used for compression mask (and MISR if present)
`compression_enable`	Control to enable compression mode
`cpi_over_npi`	Reserved for future use
`custom`	User specified control signal
`decomp_pipe_bypass`	Reserved for future use
`elastic_reset`	Reserved for future use
`first_core`	Reserved for future use

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`in_pipe_bypass`	Reserved for future use
`lbist_direct_access`	Reserved for future use
`lbist_test_enable`	Reserved for future use
`lssd_scan_clock_a`	Reserved for future use
`lssd_scan_clock_b`	Reserved for future use
`mask_enable`	Control to enable channel masking
`mask_load`	Control to enable mask loading mode
`mbist_retention_test_out`	Output test signal defined with `pmbist_use` attribute indicating `retention_pause_active`
`misr_bypass`	Reserved for future use
`misr_reset`	Reserved for future use
`misr_reset_enable`	Control signal to reset the MISR (misr compression only)
`opcg_enable`	Control to enable OPCG mode
`opcg_load`	Control to enable concatenation of the OPCG side-scan registers across cores
`opcg_load_clock`	OPCG side scan load clock
`opcg_pllclk_capture`	Control to select the clock connected to the PLLCLK input of the OPCG domain macro for capture
`opcg_trigger`	Control to enable Trigger Macro
`out_pipe_bypass`	Reserved for future use
`parallel_sdi`	Reserved for future use
`parallel_sdo`	Reserved for future use
`pipe_sen`	Control pin that enables the pipeline flops. It is usually a shift enable pin.
`scan_clock`	Reserved for future use

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`select_bypass`	Select 1500 bypass mode
`select_serial`	Select 1500 serial mode
`select_wir`	Select Wrapper Instruction Register (WIR) access control signal
`serial_sdi`	1500 Wrapper Serial Input
`serial_sdo`	1500 Wrapper Serial Output
`serial_wrck`	Clock used for inserted 1500 wrapper instruction registers (WIR) and 1500 bypass registers
`shift_enable`	Test signal that controls the shifting of data in a scan chain when asserted to its active value.
`shift_wr`	`WIR` shift control
`smartscan_broadcast`	Reserved for future use
`smartscan_enable`	Reserved for future use
`spread_enable`	Control to enable the xor decompressor
`test_clock`	Reserved for future use
`test_mode`	Test signal that controls the design in test mode
`update_wr`	`WIR` update control
`wext`	Wrapper Extest control signal
`wint`	Wrapper Intest control signal
`wir_test`	Test-mode signal to put the WIR in its ATPG-test mode
`wrapper_and_compression_clock`	Reserved for future use
`wrapper_clock`	Clock used for 1500 dedicated wrapper cells
`wrapper_reset`	Wrapper instruction register reset control

Note: Some restrictions apply for the functions.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hookup_pin {pin|hpin}`

Specifies the core-side hookup pin to be used for the top-level signal during DFT synthesis.

Note: When you specify this option, the Genus-DFT engine does not validate the controllability of any logic between the top-level signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {non_inverted | inverted}`

Specifies the polarity of the signal at the core-side hookup pin.

By default, a non inverted polarity is assumed. The Genus-DFT engine does not validate the polarity at the specified hookup pin.

`-ideal`

Marks the test bus port signal as ideal. This prevents buffering of the test bus port signal network during optimization.

Default: The test bus port signal is marked non-ideal.

Note: If the test signal is marked as ideal, Genus sets the `ideal_network` attribute to `true` on the pin or port for the test signal.

`-index integer`

Specifies the index of the test bus port.

When more than one test bus port exists with the same function, they will have different indexes. Examples are:

- In the case of an N-bit scan data input bus, each port has the `compress_sdi` function, but the index varies between `0` and `N-1`.
- In the case of `wide2` masking, two `mask_enable` test bus ports are defined, the first with an index of 0, the second with an index of 1.

Note: You cannot specify the `-index` option when the `-wir_signal` option is specified.

`-lec_value {auto | 0 | 1 | no_value}`

Specifies the approach to constraining the pin for LEC validation by the `write_do_lec` command. You can specify any of the following values:

- `auto`—Writes the opposite of the test bus port active value as the “add pin constraint” in the do file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- 0—Writes a logic `0` for the `add pin constraint` in the do file
- 1—Writes a logic `1` for the `add pin constraint` in the do file
- no_value—Does not write the signal as an “add pin constraint” in the do file

Default: `auto`

`-multi_mode`
Indicates that the test bus port signal is used for multi-mode configuration purposes.

`-name` *`string`*
Specifies the name of the test bus port.

Default: *`dft_prefix_port`*

where *`dft_prefix`* is the value of the `dft_prefix` root attribute and port is the name of the port on which the test bus port is defined.

`{`*`pin`*`|`*`hpin`*`|`*`port`*`}`
Specifies the driver for the test bus port.

`-pipeline_depth` *`integer`*
Specifies the number of pipleline stages to be inserted for this test bus port signal.

`-port_bus {`*`port`*`|`*`port_bus`*`|`*`hport_bus`*`}`
Specifies the driving port or port_bus.

`-shared_in`
Specifies whether the input port is also used as a functional port.

By default, the signal applied to the specified driving pin or port is considered to be a dedicated test signal.

This option should be specified for shared functional ports that are defined with the following function types: `async_set_reset`, `test_mode`, `shift_enable`.

Note: This option cannot be specified with the `-create_port` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Important

Specify this option for the shared test signals (such as those driving functional logic in the golden design) to ensure that these test signals will not get constrained in the `write_do_lec` dofile. Not specifying this option for a shared test signal will result in over constraining the `write_do_lec` dofile which can lead to false EQs.

Additionally, shared test signals should not be constrained in the tool-generated DFT SDC file for the NON DFT MODE mode. By omitting this option for a shared test signal, the test signal will be listed as

- a `set_false_path` constraint when defined as an async set/reset test signal
- a `set_case_analysis` constraint when defined as a shift-enable or t est-mode signal.

Either of these constraints will result in not optimizing the signal paths when the constraints are merged with the FUNCTIONAL mode.

`-test_only`

Specifies that the input port will only be used for test purposes.

`-wir_reset_value {low|high}`

Specifies the value that the test bus port will be driven to when the Wrapper Instruction Register (WIR) is reset.

You can only specify this option if you specified the `-wir_signal` option.

Default: `high` for test bus ports with the `select_serial` or `select_bypass` function, and `low` for all other functions.

`-wir_signal`

Specifies that the test bus port will be driven by a Wrapper Instruction Register (WIR) output. When you invoke the `add_wir_signal_bits` command, a bit is inserted in the WIR register for this test bus port.

Note: You can only specify this option for the following function types: `compression_enable`, `select_bypass`, `select_serial`, `spread_enable`, `wext`, `wint`, and `custom`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-wir_tm_value {low|high}`

Specifies the value that the test bus port will be forced to when the test bus port with `wir_test` function (if one exists) is driven `high` and the Wrapper Instruction Register (WIR) is ATPG-tested.

You can only specify this option with the `-wir_signal` option.

Default: the value of the `-wir_reset_value` option

Related Information

Affects this command:	insert_test_compression on page 1106
Sets these attributes:	test_bus_port Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dft_trace_back

```
dft_trace_back
     [-mode integer] [-polarity]
     [-continue] [-print]
     {subport|port|pin}
```

Returns the pin or port found by tracing back one level from the specified pin or port based on the requested mode. If a constant is encountered, the command returns 0 or 1.

This command uses the default DFT mode to perform constant propagation, but alternative modes can be selected by running `check_dft_rules -dft_configuration_mode` *`new_configuration_mode`* beforehand. Also, functional tracing can usually be done by using mode 0.

Options and Arguments

`-mode` *`integer`*

Specifies the mode for tracing back.

- 0 does not perform constant propagation. This is usually identical or close to a functional mode trace back.
- 1 performs tied-constant propagation
- 2 performs tied-constant and test-mode propagation
- 3 performs tied-constant, test-mode and shift-enable propagation . This is usually identical to the default scan shift mode.

Default: `3`

`-continue`

Specifies to continue the trace back until a primary input, complex gate, or sequential gate is reached. Additionally, the trace will terminate if a logic constant is returned for the trace back pin.

`{`*`pin`*`|`*`port`*`|`*`subport`*`}`

Specifies the pin or port from which to start the trace back.

`-polarity`

Specifies whether to report if the polarity changed through the trace.

A returned value of 0 indicates no inversion.

A returned value of 1 indicates an inversion.

`-print`

Prints the pin and polarity at every trace back.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- Following command traces back without performing constant propagation.

```
legacy_genus:/> dft_trace_back -mode 0 \
 /designs/top/instances_hier/g121/pins_in/CME
/designs/top/ports_in/cme
```

- Following command traces back using the default mode. In this case a constant logic 1 value is reported.

```
legacy_genus:/> dft_trace_back /designs/top/instances_hier/g121/pins_in/CME
1
```

- Following command traces back using the default mode and requests to report whether there was a change in polarity. In this case, a constant logic 1 with no change in logic polarity is reported.

```
legacy_genus:/> dft_trace_back /designs/top/instances_hier/g121/pins_in/CME \
 -polarity
1 0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

fix_dft_violations

```
fix_dft_violations
     { -clock -test_control test_signal
       [-test_clock_pin {pin|port} [-rise | -fall]]
     | {-async_set | -async_reset | -async_set -async_reset}
       -test_control test_signal [-async_control test_signal]
       [-insert_observe_scan
         -test_clock_pin {pin|port} [-rise | -fall]]}
     [-violations violation_object_id_list]
     [-tristate_net]
     [-xsource [-exclude_xsource instance...]]
     [-preview] [-dont_check_dft_rules] [-dont_map]
      [-design design]
```

Automatically fixes either

- *All* DFT violations of the specified types (clock, asynchronous set, asynchronous reset, tristate, or xsource).

  Only the specified violation types are fixed. If you allow to fix asynchronous set and reset violations using the same test mode signal, you can request both types to be fixed at the same time.

- The *identified* violations

  You can further limit the violations that Genus must fix to by specifying the violation ID (through the `-violations` option).

Note: Currently, clock violations are only fixed for the muxed scan style.

Options and Arguments

`-async_control` *`test_signal`*
: Specifies the name of the test signal to use to control the asynchronous violations to be fixed.

`-async_reset`
: Fixes the asynchronous reset violations on all instances.

`-async_set`
: Fixes the asynchronous set violations on all instances.

`-clock`
: Fixes the clock violations on all instances.

`-design` *`design`*
: Specifies the name of the design whose violations must be checked.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dont_check_dft_rules`

Prevents the DFT rules from being checked automatically after fixing violations.

`-dont_map`

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

`-exclude_xsource` *`instance`*

Specifies instances to exclude from automatic fixing of X-source violations.

`-insert_observe_scan`

Inserts a flip-flop for observability. If you fix DFT violations in a generic netlist, the flip-flop is mapped to a scan flip-flop during synthesis. If you fix DFT violations in a mapped netlist, the flip-flop is mapped to a scan flip-flop. You need to rerun the `check_dft_rules` command to update the DFT status of the observability flop prior to connecting it in a scan chain.

Note: This option can only be used when fixing an asynchronous set reset violation and requires the `-test_clock_pin` option.

{*`pin`* | *`port`*}

Specifies the test signal pin or port to use to control the set or reset. By default, the set or reset are controlled by the `-test_control` option.

To specify this option, the pin or port must first be declared as a `test_mode` signal using the `define_dft test_mode` constraint.

`-preview`

Reports how the violations will be fixed, without making modifications to the netlist.

`[-rise | -fall]`

Specifies to use the rising or falling edge of the test clock to fix the DFT violation during test-mode operation.

Default: `-rise`

`-test_clock_pin` {*`pin`* | *`port`*}

Specifies the test clock signal to be used. Specify the pin or port from where the clock signal originates. In most applications, the clock pin or port is identified when checking the DFT rules.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This option is optional when fixing clock violations. By default, the Genus-DFT engine performs a clock trace to identify a controllable test clock that appears in the fanin cone of the clock violation and uses this test clock to fix the actual clock violation.

This option is required when you want to insert observability flip-flops when fixing async violations. In this case, the test clock signal drives the clock pin of the observation flip-flops during test-mode operation.

`-test_control` *`test_signal`*

Specifies the name of the test signal to use to fix the violation.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

`-tristate_net` Specifies to fix tristate net contention design rules violations.

`-violations` *`violation_object_id_list`*

Fixes the violations that are identified by their object name.

Note: The `check_dft_rules` command creates a `violations` directory in the design hierarchy under `/designs/`*`design`*`/dft/report`. The objects in this directory correspond to the violations found during the last execution of the `check_dft_rules` command. Use the `report dft_violations` command to list all remaining violations in the design.

`-xsource` Specifies to fix X-Source design rules violations.

Examples

- The following example instructs Genus to fix all clock, async set, and asynch reset violations using test mode signal `tm` and test clock `CK1` using the rising edge as the active edge.

```
fix_dft_violations -clock -async_set -async_reset
-test_control tm -insert_observe_scan -test_clock_pin CK1 -rise
```

If you do not want to use the same test clock to fix the clock violations and to drive the clock pin of the observation flip-flops, you need to enter two commands. For example,

```
fix_dft_violations -clock -test_control tm
```

```
fix_dft_violations -async_set -async_reset -test_control tm
-insert_observe_scan -test_clock_pin CK1 -rise
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

In this case, the Genus-DFT engine automatically determines which test clock to use to fix the clock violations.

- The following example instructs Genus to fix all clock, async set, and asynch reset violations using test mode signal `tm` and test clock `CK1` using the rising edge as the active edge.

```
fix_dft_violations -clock -async_set -async_reset
-test_control tm -test_clock_pin CK1 -rise
```

- The following example instructs Genus to fix violations `vid_1` and `vid_3` if they are violations of type `async_set`.

```
fix_dft_violations -violations {vid_1 vid_3} -async_set -test_control tm
```

Related Information

Affected by these constraints:	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995
Affects these commands:	check_dft_rules on page 870
	report dft_registers on page 1128
	report dft_violations on page 1130
	syn_map on page 529
Sets these attributes:	dft_status
	dft_violation
	Violations Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

fix_scan_path_inversions

`fix_scan_path_inversions` *`actual_scan_chain`*`...`

Fixes inversions for every scan element in the scan path. The command inserts inverters as required in a scan chain to remove inversions along the scan data path.

Options and Arguments

`actual_scan_chain`

Specifies the scan chain(s) to be analyzed and to insert inverters.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

identify_domain_crossing_pins_for_cgic_and_scan_abstracts

```
identify_domain_crossing_pins_for_cgic_and_scan_abstracts
     [-cgic] [-bbox] [-add_fencing]
     [-global_blocking_shift_enable test_signal][design]
```

Identifies domain crossing pins for blackboxes and integrated clock-gating instances in the specified design and adds blocking logic. This is needed as Modus ATPG assumes by default that all clock pulses in the same cycle have their rising edges aligned, whereas in reality due to different speeds and delays this is not true especially in cases where the enable logic for a clock is controlled by enable logic from a different clock. This can lead in simulation mis-compares where the clock propagation stops in for example, a CGIC. By gating the CGIC Test Enable pin, we make sure that clocks are free to propagate as Modus expects.

Options and Arguments

`-add_fencing`	Specifies to add fencing for the identified domain crossing pins using the global Shift Enable.
`-bbox`	Specifies to identify domain crossing pins for blackboxes.
`-cgic`	Specifies to identify domain crossing pins for integrated clock-gating instances, and potentially gate their Test Enable pins if -`add_fencing` is also used.
`design`	Specifies the design in which you want to identify domain crossing pins.
`-global_blocking_shift_enable` *`test_signal`*	Specifies the global blocking shift enable signal.

Related Information

Related command: insert_dft opcg on page 1063

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

identify_multibit_cell_abstract_scan_segments

```
identify_multibit_cell_abstract_scan_segments
     [-dont_check_dft_rules] [-preview]
     [-libcell libcell...] [-design design]
```

Identifies multi-bit scan cells in the design, and defines scan abstract segments for each instance of the multi-bit scan cell.

Multi-bit scan cells implemented using either a parallel or serial bit approach are supported by the tool.

Refer Mapping to Multibit Scan Cells in th Other Flows Chapter in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-design` *design*	Specifies the name of the top-level design on which to identify abstract segments for multi-bit scan cells. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
`-dont_check_dft_rules`	Prevents the DFT rules from being automatically checked after identifying abstract segments for multi-bit scan cells.
`-libcell` *libcell*	Specifies the multi-bit library cells on which to perform abstract segment identification.
`-preview`	Reports the identified abstract scan segments without defining them.

Examples

- The following example identifies a parallel multi-bit scan cell with two bits; where each bit would be defined as an abstract scan segment:

  ```
  legacy_genus:/> identify_multibit_cell_abstract_scan_segments -preview
  ```

  Would execute command:

  ```
  define_dft abstract_segment -length 1 -sdi SI1 -sdo Q1
  -shift_enable_port SE1 -active high -clock_port CP -rise
  -libcell/libraries/tcbn65ulp_c070701wc2/libcells/DUALSDFQD0 -name DUALSDFQD0
  ```

  Would execute command:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
define_dft abstract_segment -length 1 -sdi SI2 -sdo Q2
-shift_enable_port SE2 -active high -clock_port CP -rise
-libcell /libraries/tcbn65ulp_c070701wc2/libcells/DUALSDFQD0 -name DUALSDFQD0
```

- The following example identifies a serial multi-bit scan cell of length 4; where each instance of the cell would be defined as an abstract scan segment:

  ```
  legacy_genus:/> identify_multibit_cell_abstract_scan_segments -preview
  ```

  Would execute command:

  ```
  define_dft abstract_segment -length 4 -sdi SI -sdo Q4
  -shift_enable_port SM -active high -clock_port CK -rise
  -libcell /libraries/cs60ale_uc_scan/libcells/YSDM4ALU1 -name YSDM4ALU1
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

identify_shared_wrapper_cells_in_design

`identify_shared_wrapper_cells_in_design [design]`

Identifies shared wrapper flops in a mapped design. The shared wrapper flops are marked for exclusion from shift register identification and multibit merging.

Note: You must set `dft_identify_shared_wrapper_cells` attribute to `true` before you run this command.

The command will set the `dft_shared_wrapper_flop` attribute set to `true` on the shared wrapper flops.

Options and Arguments

design	Specifies the name of the design for which to identify the wrapper flops. This argument is required if multiple designs are present.

Related Information

Affected by this attribute:	dft_identify_shared_wrapper_cells
Sets this attribute:	dft_shared_wrapper_flop

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

identify_shift_register_scan_segments

```
identify_shift_register_scan_segments
     [-preview] [-incremental] [-design design]
```

Identifies all shift registers in the design.

The Genus-DFT engine uses the following naming convention for automatically identified shift-register segments:

`DFT_AutoSegment_n`

You can find the automatically identified shift-register segments in:

`/designs/top_design/dft/scan_segments`

Note: A shift register segment contains flops driven by the same clock and same clock edge.

A shift register is a scan segment which can be associated with either

- A user-defined top-level chain—created using the `define_dft_scan_chain` command
- A tool-created scan chain—created using the `connect_scan_chains` command

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-design design`	Specifies the name of the design in which the shift registers must be identified. If you omit the design name and multiple designs are loaded, the top-level design of the current directory of the design hierarchy is used.
`-incremental`	Adds new shift register segments in incremental mode, without changing already identified shift register segments stored in `/designs/design/dft/scan_segments` **Note:** Without this option, the existing identified shift register segments will be removed and new segments will be identified based on the new values specified for the command options.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-preview`	Reports which shift register segments will be identified, without adding them to the list of scan segments.

Related Information

Sets this attribute: user_defined_segment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

identify_test_mode_registers

```
identify_test_mode_registers
     { -fixed_value_register_file file
     | {-stil file [-macro string] | -mode_init file}
       [-library string] [-atpg_log string]
       [-continue_with_severe_warnings] }
     [-design design] [-preview]
```

Identifies all internal registers whose output signals must remain constant during test mode and generates the corresponding test-mode signals required for the Genus-DFT engine.

The fixed-value registers can be identified

- From an existing ATPG log file
- By invoking Modus and simulating a mode initialization sequence

  Note: In the latter case, you need to have the Modus software installed and your operating system `PATH` environment variable must include the path to the Modus software.

This command is not generally used and is intended for a custom flow where a special test mode initialization sequence is executed in Modus to identify all fixed-value registers that hold their value constant during the entire test mode.

The fixed-value registers and their constant values can be fed back to a Genus DFT synthesis session to simplify the DFT setup. The alternative is for the user to manually identify all internal test mode fixed-value registers.

Options and Arguments

`-atpg_log` *`string`*
: Specifies the log file for Modus.

`-continue_with_severe_warnings`
: Continues the Modus run even when severe warnings are issued.

`-design` *`design`*
: Specifies the name of the design for which to define the test-mode signals.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-fixed_value_register_file` *`file`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Specifies an ATPG log file from a previous run that contains a list of fixed value registers. When this option is specified, Genus can parse this file instead of invoking Modus to get the list of fixed value registers. **Note:** This option cannot be specified with the `-stil`, `-mode_init`, `-macro`, `-library`, `-atpg_log` and `-continue_with_severe_warnings` options.
`-library` *`string`*	Specifies the list of Verilog structural library files. Specify the list in a quoted string. Refer to the `write_et_atpg` `-library` option description for additional information. **Note:** This option is only required when you invoke this command on a mapped netlist.
`-macro` *`string`*	Specifies the name of the macro in the STIL file that contains the initialization vectors to be simulated. *Default*: `test_setup`
`-mode_init` *`file`*	Specifies the name of the file that contains the mode initialization sequence in TBDpatt format.
`-preview`	Reports the fixed value registers but does not set the test mode values.
`-stil` *`file`*	Specifies the name of the STIL file that contains the mode initialization sequence.

Examples

- The following command creates a report of the list of the registers with fixed values.

```
legacy_genus:/> identify_test_mode_registers -stil newStil -prev
Identifying internal registers with fixed value outputs under test_mode setup.
... Creating intermediate files

WARNING : No user defined shift enable signal found.
ATPG interface file may contain incomplete information
Cadence Design Systems RC file: Cadence ATPG file created successfully.

----------------------------------------
... Identifying the fixed value registers
----------------------------------------

Test Function    Block Name
+TI              tc/ts_reg[0]
+TI              tc/ts_reg[1]
+TI              tc/ts_reg[2]
----------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Note:  The +/-TI Test Function flag denotes the active logic value that a signal
is to be held to during test mode.
       +TI denotes a logic value 1; -TI denotes a logic value 0
```

- The following command creates the test signals and automatically reruns the DFT rule checker.

```
legacy_genus:/> identify_test_mode_registers -stil newStil
Identifying internal registers with fixed value outputs under test_mode setup.
... Creating intermediate files

WARNING : No user defined shift enable signal found.
ATPG interface file may contain incomplete information
Cadence Design Systems RC file: Cadence ATPG file created successfully.

----------------------------------------
... Identifying the fixed value registers
----------------------------------------
INFO: Setting active high test mode signal on tc/ts_reg[0]/q
INFO: Setting active high test mode signal on tc/ts_reg[1]/q
INFO: Setting active high test mode signal on tc/ts_reg[2]/q
----------------------------------------
  Checking DFT rules for 'top' module under 'muxed_scan' style

...
legacy_genus:/> ls dft/test_signals
/designs/top/dft/test_signals:
./                          rst                         tc__ts_reg[1]__q
incr                        tc__ts_reg[0]__q            tc__ts_reg[2]__q
```

Related Information

Affected by these constraints:	define_dft test_mode on page 995
	define_dft test_clock on page 990
Sets this attribute:	user_defined_signal

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft

```
insert_dft
     { boundary_scan | compression_logic
     | jtag_macro  | lockup_element
     | logic_bist | opcg
     | ptam |  pmbist | scan_power_gating
     | shadow_logic | shift_register_test_points
     | test_point | user_test_point | wrapper_cell
     | wrapper_instruction_register
     | wrapper_mode_decode_block}
```

Inserts DFT test logic.

Options and Arguments

`boundary_scan`	Inserts boundary scan cells and the corresponding JTAG controller.
`compression_logic`	Inserts compression logic in a design that requires a fixed number of scan compression channels.
`jtag_macro`	Inserts a JTAG Macro controller into a netlist.
`lockup_element`	Inserts lockup elements in the specified analyzed scan chains.
`logic_bist`	Inserts Logic Bist (LBist) logic into the design.
`opcg`	Inserts OPCG logic that generates on-chip launch and capture clocks for testing of at-speed delay defects.
`ptam`	Inserts Power Test Access Mechanism (PTAM) control logic into the design.
`pmbist`	Inserts Programmable Memory Built-In-Self-Test (PMBIST) logic in the design for memories based on the definitions in the configuration file specified using the `-config_file`.
`scan_power_gating`	Inserts gating logic at selected flop outputs to minimize switching power during scan shift
`shadow_logic`	Inserts DFT shadow logic to enable testing of shadow logic around a module.
`shift_register_test_points`	Creates a shift register by inserting test points.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`test_point` Inserts a native test point.

`user_test_point` Inserts a user-defined test point.

`wrapper_cell` Inserts an IEEE-1500 style core-wrapper cell.

`wrapper_instruction_register`

Inserts a Wrapper Instruction Register (WIR) of length 3 and connects the Wrapper Control Signals, serial scan-in and serial scan-out to the WIR.

`wrapper_mode_decode_block`

Builds a 1500 mode decode block based on the scan modes that were defined with type `wrapper`.

Related Information

Related commands:

insert_dft boundary_scan on page 1026
insert_dft jtag_macro on page 1046
insert_dft lockup_element on page 1050
insert_dft logic_bist on page 1051
insert_dft opcg on page 1063
insert_dft pmbist on page 1065
insert_dft ptam on page 1071
insert_dft scan_power_gating on page 1074
insert_dft shadow_logic on page 1077
insert_dft shift_register_test_points on page 1082
insert_dft test_point on page 1083
insert_dft user_test_point on page 1089
insert_dft wrapper_cell on page 1094
insert_dft wrapper_instruction_register on page 1102
insert_dft wrapper_mode_decode_block on page 1104

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft boundary_scan

```
insert_dft boundary_scan [-design design]
     [-comp_enables_high port [port]...]
     [-comp_enables_low port [port]...]
     [-exclude_ports port [port]...]
     [-functional_clocks port [port]...]
     [-custom_cell_directory string]
     [-bcells_location instance]
     [-jtag_macro_location instance]
     [-generate_wrck]
     [-pinmap_file file | -physical]
     [-power_on_reset pin|port]
     [-tck port] [-tdi port] [-tdo port]
     [-tms port] [-trst port]
     [-dont_map] [-preview]
     [-preserve_tdo_connection] [-skip_checking_tap_port_iocell]
```

Inserts boundary scan cells and the corresponding JTAG Macro (if it is not yet instantiated in the design).

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-bcells_location` *`instance`*

Specifies the instance in which to insert the boundary scan cells. By default, the boundary scan cells are inserted in the same hierarchy as their respective pad cells in the design.

`-comp_enables_high (-comp_enables_low)` *`port...`*

Specifies that the compliance value for the specified ports is active high (low) during functional mode. The compliance enable value is the value that a test port (test mode, shift enable) is tied to during the functional mode of the chip.

The ports with their compliance value are added to a BSDL `COMPLIANCE_PATTERNS` statement.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-custom_cell_directory` *`string`*

Specifies the path to the directory that contains the files describing the custom boundary cells. Each cell must be described as a Verilog module in its own file. The basename of the file must match the name of the cell in the module description.

`-design` *`design`*

Specifies the name of the design in which you want to insert boundary scan logic. This option is required if you have loaded multiple designs.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dont_map`

Prevents the inserted boundary scan logic from being mapped to technology gates even if the design is already mapped.

`-exclude_ports` *`ports`*

Excludes the specified ports from being considered for boundary scan logic insertion.

`-functional_clocks` *`ports`*

Specifies the ports that are seen as clocks for ATPG. These ports include

- async set and reset ports
- clocks used in functional mode, but not in scan shift mode

`-jtag_macro_location` *`instance`*

Specifies a hierarchical instance into which to insert the JTAG macro. By default, the JTAG macro is inserted in the top-level design.

`-generate_wrck`

Creates a `JTAG_WRCK` output pin on the JTAG macro. This output port gates the test clock of the JTAG macro (TCK) for all JTAG FSM states other than `Run-Test-Idle`, `CAPTUREDR`, `SHIFTDR`, or `UPDATEDR`.

The `JTAG_WRCK` pin can be used to facilitate interfacing an IEEE 1500 wrapper serial control (WSC) bus to a JTAG macro.

`-no_trst_test_signal`

Prevents defining an active-low test signal on the TRST port and causes a formal verification constraint with a value of ‘0’ to be added for the TRST port for the revised design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-physical`

Specifies to build the boundary scan register using physical information to minimize its wire length.

The physical placement information is obtained from the DEF file read in with the `read_def` command

Note: This option is mutually exclusive to the `-pinmap` option.

`-pinmap_file` *`file`*

Specifies the name of the file containing the mapping between the design ports and the actual package pins. This mapping is also used to determine the boundary scan order.

Note: This option is mutually exclusive to the `-physical` option.

`-power_on_reset {`*`pin`* | *`port`*`}`

Specifies the power-on-reset pin which will be connected to the `JTAG_POR` input pin on the `JTAG_Macro` subdesign. This connection is made when the boundary scan logic is inserted in the design.

Note: If the power-on-reset signal is applied to a top-level port, you also need to exclude this port from inclusion into the boundary scan register using the `-exclude_ports` option.

`-preserve_tdo_connection`

Preserves the existing net connection to from-core and tristate enable pins of the TDO pad cell.

If you do not specify this option, the existing net connections will be broken and new net connections will be made during boundary scan insertion from the `JTAG_TDO` and `JTAG_ENABLE_TDO` pins to the from-core and tristate enable pins of the TDO pad cell, respectively.

Note: If you preserve the TDO connections, such that the net is driven by user logic other than a pre-instantiated JTAG macro, boundary scan insertion will insert a JTAG macro and leave its `JTAG_TDO` pin unconnected in the netlist.

`-skip_checking_tap_port_iocell`

Prevents the tool from checking for the pad cell on TAP ports in case of block-level insertion,

`-preview`

Shows the potential changes, without making any modifications to the netlist.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-tck` *`port`*	Specifies the port name of the driver for the test clock of the JTAG macro. Specify this option when the existing JTAG port does not use the standard name. *Default*: `TCK`
`-tdi` *`port`*	Specifies the port name of the driver for the test data (scan) input of the JTAG macro. Specify this option when the existing JTAG port does not use the standard name. *Default*: `TDI`
`-tdo` *`port`*	Specifies the port name of the test data (scan) output of the JTAG macro. Specify this option when the existing JTAG port does not use the standard name. **Note:** An existing TDO port must have a tristate I/O pad. *Default*: `TDO`
`-tms` *`port`*	Specifies the port name of the test mode select input of the JTAG macro. Specify this option when the existing JTAG port does not use the standard name. *Default*: `TMS`
`-trst` *`port`*	Specifies the port name of the (asynchronous) test reset of the JTAG macro. Specify this option when the existing JTAG port does not use the standard name. *Default*: `TRST`

Related Information

Related constraints:	define_dft jtag_instruction_register on page 937
	define_dft jtag_macro on page 939
	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995
Affects these commands:	compress_scan_chains on page 882
	insert_dft ptam on page 1071
Sets these attributes:	boundary_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dft_jtag_macro_exists

JTAG Port Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft compression_logic

```
insert_dft compression_logic [design]
     -use_existing_channels actual_scan_chains
     {-use_user_scan_chains scan_chains
     |-build_new_scan_chains integer}
     [-allow_shared_clocks] [-auto_create]
     [-decompressor {broadcast
                    | xor [-spread_enable test_signal]}]
     [-master_control test_signal]
     [-compression_enable test_signal] [-target_period integer]
     [ -compressor xor
          [-mask {wide1|wide2} [-mask_clock {port|pin}]
          [-mask_load test_signal]
          [-mask_enable test_signal]
          [-mask_sharing_ratio integer]
          [-apply_timing_constraints
            [-timing_mode_names mode_list]]
          [-write_timing_constraints file]
          [-low_pin_compression
            [-lpc_control shift_enable]
            [-shift_enable shift_enable]]
     | -compressor misr
          [-serial_misr_read
          [-misr_observe test_signal]]
          [-misr_clock {port|pin}]
          {-misr_reset_enable test_signal
          | -misr_reset_clock test_signal}
          [-misr_read test_signal
          | -use_all_scan_ios_unidirectionally]
          [-misr_shift_enable test_signal]
          [-mask_sharing_ratio integer]
          [-mask {wide0 | wide1 | wide2}]
          [-mask_clock {port|pin}] [-mask_load test_signal]
          [-mask_enable test_signal_list]
     | -compressor hybrid
          [-serial_misr_read]
          [-misr_observe test_signal]
          [-misr_bypass test_signal]
          [-misr_clock {port|pin}]
          {-misr_reset_enable test_signal
          | -misr_reset_clock test_signal}
          [-misr_shift_enable test_signal]
          [-mask {wide0 | wide1 | wide2}]
          [-mask_sharing_ratio integer]
          [-mask_clock {port|pin}] [-mask_load test_signal]
          [-mask_enable test_signal_list] }
     [-dont_exploit_bidi_scanio] [-inside instance] [-preview]
     [-jtag_control_instruction jtag_instruction]
          [-allow_multiple_jtag_control]]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Inserts compression logic in a design that requires a fixed number of scan compression channels. It concatenates these channels into the specified number of full scan chains, and adds the information about the newly created fullscan chains to the `actual_scan_chains` directory while removing the original compression channels from the same directory.

Prerequisite: The design must have connected compression channels and these channels must appear in the `actual_scan_chains` directory. The channels should not be compressed (the `compressed` attribute must be `false`) and all elements in the compression channels must have passed the DFT rule checks. These compression channels should also not appear as scan chain definitions in the `scan_chains` directory.

Tip

The scan data input (sdi) pin and the scan data output (sdo) pin of the original compression channels will be disconnected but will be left in the netlist. It is the user's responsibility to remove them if desired.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-allow_multiple_jtag_control`

When controlling the compression testmode from a JTAG macro, (that is, the `-jtag_control_instruction` option is specified), `compress_scan_chains` checks for the presence of other compression macros that are also JTAG controlled. RC-DFT does not automatically support such a configuration and therefore insertion of a second JTAG controlled compression macro is disallowed by default. Specify this option to bypass this check and insert additional JTAG controlled compression macros. When this option is specified, the tool assumes you will manually perform any additional stitching needed and will appropriately modify any files generated for Modus.

Note: You must also specify the `-jtag_control_instruction` option with this option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-allow_shared_clock`

Allows the mask or `MISR` clock to be shared with an existing full-scan test clock.

Note: The mask or `MISR` clock can only be shared with a test clock if you added gating logic which prevents the scan flops from pulsing during the channel mask load or `MISR` reset sequences. Since functional clocks are typically used for scanning, this requirement means that the functional clocks must be gated during test.

Important

When specified with the `-auto_create` option, a `-mask_load` pin is automatically created. The mask_load pin must additionally be used to gate off the clock being shared. If this gating logic is not added, the mask loading or MISR reset procedure will corrupt the test data in the design.

`-apply_timing_constraints`

Applies timing constraints to the appropriate compression control signals to prevent the mapper from considering these paths for timing optimization.

Timing constraints will be applied in all user-specified timing modes.

Note: If your design has multiple timing modes but you did not specify the `-timing_mode_names` option to list the timing modes for which to write the constraints, no additional constraints are applied.

`-auto_create`

Automatically creates the necessary test pins as top-level ports.

If you omitted any of the following options `-compression_enable`, `-spread_enable`, `-mask_clock`, `-mask_load`, `-mask_enable`, `-mask_or_misr_sdi`, `-mask_or_misr_sdo`, `-misr_clock`, `-misr_observe`, `-misr_reset_enable`, `-misr_read`, `-misr_bypass`, the appropriate ports will be created and named using the following format: *`prefixOption`*

For example, *`prefix`*`compression_enable`, where *`prefix`* is the value of the `dft_prefix` root attribute.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-build_new_scan_chains` *`integer`*

Specifies the total number of top-level scan chains that is desired.

By default, the new ports will be called `SDI[`*`i`*`]` and `SDO[`*`i`*`]`, while the new chains will be called `CHAIN_`*`i`*, where *`i`* starts from 0 if no other ports or chains with those names exist.

`-compression_enable` *`test_signal`*

Specifies the name of the test signal that enables configuring the actual scan chains in compression mode.

Note: If you do not specify the `-auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. If you request to build the compression logic with a master control signal (`-master_control`), the input port driving the compression enable signal can be an existing functional pin (specified through the `-shared_in` option of `define_dft test_mode`). If you do not specify the `-master_control` option, you must define the compression enable signal without the `-shared_in` option.

`-compressor {xor | misr| hybrid}`

Specifies the type of compression logic to be built:

- `xor` specifies to build an XOR-based compressor
- `misr` specifies to build a `MISR-based` compressor
- `hybrid` specifies to build a `MISR` compression with `MISR` bypass capability. Bypassing the MISR allows you to perform compression using just the XOR compressor.

Default: `xor`

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic to be built:

- `xor` specifies to build an XOR-based spreader network in addition to the broadcast-based decompression logic
- `broadcast` specifies to build a broadcast-based decompression logic (simple scan fanout).

Default: `broadcast`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

design

Specifies the name of the top-level design whose scan chains must be compressed. You should specify this name in case you have multiple top designs loaded.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-inside` *instance*

Specifies the instance in which to instantiate the compression logic.

By default, the compression logic is inserted as a hierarchical instance in the top-level of the design.

`-jtag_control_instruction` *jtag_instruction*

Specifies which JTAG instruction will be used to target the compression macro's test data register.

`-lpc_control` *shift_enable_signal*

Specifies a shift-enable signal used to control low pin count compression.

You cannot specify the same shift-enable signal for both the `-lpc_control` and `-shift_enable` options.

If you omit this signal, the tool will use one of the default shift-enable signals.

If no shift-enable pin exists, the tool creates an LPC_CONTROL shift-enable pin if the `-auto_create` option is specified.

`-low_pin_compression`

Reduces the number of compression control pins required by using encoded control signals.

`-mask {wide0| wide1 | wide2}`

Inserts scan channel masking logic of the specified type.

The masking types that can be used depend on the compressor type specified with the `-compressor` option.

By default, no masking logic is inserted.

Note: The syntax indicates which types are available for each of the compressor types.

`-mask_clock {`*pin*`|`*port*`}`

Specifies the clock that controls the mask registers.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: The input port associated with this option can be an existing functional pin. This clock cannot be shared with an existing full-scan test clock pin unless you also specify the `-allow_shared_clocks` option.

`-mask_enable` *`test_signal`*

Specifies the name of the test signal that controls whether mask bits should be applied during the current scan cycle.

For `wide2` masking, two mask enable signals must be specified.

Note: If you do not specify the `-auto_create` option, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-mask_load` *`test_signal`*

Specifies the name of the test signal that enables loading of the mask data into the mask data registers.

Note: If the mask_clock is dedicated (that is, is only used for mask register loading), this signal is not needed. If this signal is shared (is used to clock the MISR or other logic in the circuit), this signal is needed to gate non mask load clock pulses from corrupting the mask registers. If the `mask_clock` is shared with other logic, you can use this signal to protect the shared logic from corruption during the mask load sequence.

`-mask_sharing_ratio` *`integer`*

Specifies the number of internal scan channels sharing a mask register. The specified integer may not exceed the value specified for the compression ratio.

Note: This option is only valid with `wide1` and `wide2` masking.

`-master_control` *`test_signal`*

Specifies the master control signal that gates the compression enable signal used for compression.

Note: This test signal must be dedicated for test and must have been defined using the `define_dft test_mode` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-misr_bypass` *`test_signal`*

Specifies the test signal used to bypass the `MISR-based` logic. This test signal is required in hybrid compression mode.

Note: If you do not specify the `-auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-misr_clock {`*`pin`*`|`*`port`*`}`

Specifies the clock that controls the `MISR` registers.

Note: This input port cannot be shared with an existing full-scan test clock unless it is only used to accumulate the MISR signature or if the `-allow_shared_clocks` option is specified. The `-misr_clock` option is only used to accumulate the MISR signature if there are separate `-mask_clock` and `-misr_reset_clock` signals specified.

`-misr_observe` *`test_signal`*

Specifies the test signal used to select Serial MISR Read. This is required when the `-serial_misr_read` option is specified unless `-auto_create` or `-jtag_control_instruction` is also specified.

Note: You must also specify the `-serial_misr_read` option with this option.

`-misr_read` *`test_signal`*

Specifies the test signal to configure any bidirectional scan I/O pads for MISR compression.

Note: This option is mutually exclusive with the `-use_all_scan_ios_unidirectionally` option. Using scan I/O bidirectionally during MISR compression is only available with the -compressor misr option. When using the `-compressor hybrid` option, all scan I/O are used unidirectionally.

`-misr_reset_clock` *`test_signal`*

Specifies a separate dedicated test signal that is used to asynchronously reset the MISR.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This option is mutually exclusive with the `-misr_reset_enable` option.

`-misr_reset_enable` *`test_signal`*

Specifies the test signal used to reset the `MISR` registers.

Notes:

- This option is mutually exclusive with the `-misr_reset_clock` option.
- If you do not specify the -`auto_create` option, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-misr_shift_enable` *`test_signal`*

Specifies the test signal used to enable MISR accumulation during scan shifting.When this signal is de-asserted, the contents of the MISR register will not change.

Note: If this option is omitted, the default shift-enable test signal for the design is used. If this option is specified, you must have defined this test signal using the `define_dft shift_enable` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft shift_enable` command).

`-preview`

Reports the requested ratio, the maximum original scan chain length, the maximum subchain length, and the number of internal scan channels that would be created without making modifications to the netlist. Use this option to verify your compression architecture prior to inserting the compression logic.

`-serial_misr_read`

Specifies to include support for reading MISR bits serially through the scan data pins.

`-shift_enable` *`shift_enable`*

Specifies the shift-enable signal to be used for low pin count compression.

If you omit this option, the tool will use the default shift-enable signal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-spread_enable` *`test_signal`*

Specifies the name of the test signal that enables applying the input test data to an XOR-based spreader network.

Use this option when `-decompressor` is set to `xor`.

Note: If you do not specify the `-auto_create` or `-jtag_control_instruction` options, this test signal must have been defined using the `define_dft test_mode` command. The input port driving this test signal can be an existing functional pin (specified through the `-shared_in` option of the `define_dft test_mode` command).

`-target_period` *`integer`*

Specifies a target clock period (in picoseconds) used to optimize the compression macro. If the value zero (`0`) is specified, synthesis is performed with a low effort compile, and without applying external (input/output delay) constraints.

Default: value for `test_clock` period of the scan chains being compressed

`-timing_mode_names` *`mode_list`*

Specifies the timing modes for which to generate the additional timing constraints that apply to the compression control signals.

The timing modes are taken into account when you specify the `-apply_timing_constraints` and `-write_timing_constraints` options.

Note: This applies only to multi-mode designs. Modes are created with the `create_mode` command.

`-use_all_scan_ios_unidirectionally`

Disables use of bidirectional scan I/O for a `MISR-based` compressor.

Note: This option is mutually exclusive with the `-misr_read` option.

`-use_existing_channels` *`actual_scan_chains`*

Specifies the names of the actual scan chains to be used as compression channels.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-use_user_scan_chains` *`scan_chains`*

Specifies which scan chain definitions (from the `scan_chains` directory) to use when building the full scan chains.

In this case, the tool will use the port and chain names defined with the `define_dft scan_chain` commands.

`-write_timing_constraints` *`file`*

Specifies the file to which to write the timing constraints applied to the appropriate compression control signals to prevent the mapper from considering these paths for timing optimization. The timing constraints are not written if you specify the `-preview` option.

Timing constraints will be applied in all user-specified timing modes.

Note: If your design has multiple timing modes but you did not specify the `-timing_mode_names` option to list the timing modes for which to write the constraints, constraints for all modes are written to the file.

Examples

In the following examples, assume that the design has 55 existing scan chains, but only five top-level scan chains are desired.

- The following command inserts compression logic in the design and auto creates the compression channels.

```
insert_dft compression_logic -build_new_scan_chains 5 -auto_create
```

- In the following example, the user first defines the five top-level chains. Next these chains are used by the insert_dft compression_logic command.

```
legacy_genus:/> define_dft scan_chain -name DFTChain1 -sdi SI1 -sdo SO1 -create_ports
legacy_genus:/> define_dft scan_chain -name DFTChain2 -sdi SI2 -sdo SO2 -create_ports
legacy_genus:/> define_dft scan_chain -name DFTChain3 -sdi SI3 -sdo SO3 -create_ports
legacy_genus:/> define_dft scan_chain -name DFTChain4 -sdi SI4 -sdo SO4 -create_ports
legacy_genus:/> define_dft scan_chain -name DFTChain5 -sdi SI5 -sdo SO5 -create_ports
legacy_genus:/> llength [find / -actual_scan_chain *]
55
legacy_genus:/> insert_dft compression_logic -use_user_scan_chains \
[find / -scan_chain DFTChain*] -auto_create
legacy_genus:/> llength [find / -actual_scan_chain *]
5
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands:	connect_scan_chains on page 906
Affects this command:	report dft_chains on page 1125
Sets these attributes:	compressed
	dft_compression_signal
	dft_mask_clock
	dft_misr_clock
	type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft hard_repair

```
insert_dft hard_repair
     -config_file config_file
     [ -connect_to_jtag | -dont_create_hri_ports
     | -direct_access_only [-dont_create_hri_ports]]
     [-dont_scan]
     [-alt_dft_cfg_mode mode]
     [-dft_cfg_mode mode]
     [-schedule_instruction string] [-allow_unused_instructions]
     [-check_instruction string]
     [-test_plan_instruction string]
     [-constraint_instruction string]
     [-repair_instruction string]
     [-directory directory][design]
```

Inserts hard repair for the PMBIST fuse control unit (FCU) for memories based on the repair channel definitions specified in the configuration file (specified with the `-config_file` option).

Support is currently limited to block-level repair channel binding.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*

Options and Arguments

`-allow_unused_instructions`

Enables the creation of all possible `jtag_instruciton_decode*` ports for PMBIST instructions for the block being processed by this command. This option assures a common interface on the block (independent of the functions enabled within this block).

`-alt_dft_cfg_mode` *mode*

Specifies the configuration mode in which the PMBIST FCU will be operating for non-JTAG mode of operation.

PMBIST FCU test pin verification must be done in this mode as well.

Default: `pmbist`

`-check_instruction` *string*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the `FCUCHK` instruction that must be loaded into the IEEE 1149.x TAP controller to check the results of PMBIST FCU execution.

Default: `FCUCHK`

`-config_file` *`config_file`*

Specifies the file that contains the user-defined configuration parameters which selects the PMBIST repair channel features and controls for the insertion of the FCU for targeted non-volatile memories.

`-connect_to_jtag`

Connects PMBIST logic's JTAG interface pins to a pre-instantiated JTAG macro (either from Cadence or a Third party).

Use this option when the design is the top level of the chip, and you have instantiated the JTAG macro and will use it to access the PMBIST FCU engines.

`-constraint_instruction` *`string`*

Specifies the `FCUAMR` instruction that must be loaded into the IEEE 1149.x TAP controller to access the FCU programmable algorithm memory and test conditions.

Default: `FCUAMR`

`design`

Specifies the name of the top-level design.

`-dft_cfg_mode` *`mode`*

Specifies the configuration mode in which the PMBIST FCU will be operating.

PMBIST FCU test pin verification must be done in this mode as well.

Default: `pmbist`

`-direct_access_only`

Specifies whether to insert PMBIST FCU without the support from the JTAG Macro. The memory repairs are controlled and examined directly from the PMBIST FCU direct access ports or pins without using the JTAG macro. In this case, any information for the JTAG macro is ignored even when a JTAG macro is specified in the design.

`-directory` *`directory`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the directory in which PMBIST must create all intermediate FCU files for its internal use.

Default: `current_working_directory/hri_design`

`-dont_create_hri_ports`

Prevents creation of PMBIST FCU control ports for subsequent JTAG macro and direct access connection at the top-level instance.

Use this option when the instance is the top level of the chip, and you have not yet instantiated the JTAG macro, or you use the direct access method and these ports were predefined.

`-dont_scan`

Allows the insertion of non-scannable PMBIST FCU logic.

Use this option if ATPG is not required for the PMBIST FCU logic or for the design. This implies that test signals do not need to be specified for the PMBIST FCU logic.

`-repair_instruction` *`string`*

Specifies the JTAG instruction that must be loaded into the IEEE 1149.x TAP controller for repair register access.

Default: `FCURR`

`-schedule_instruction` *`string`*

Specifies the `FCUSCH` instruction that must be loaded into the IEEE 1149.x TAP controller to schedule repair channels for PMBIST FCU execution.

Default: `FCUSCH`

`-test_plan_instruction` *`string`*

Specifies the `FCUTPN` instruction that must be loaded into the IEEE 1149.x TAP controller for test plan selection during PMBIST FCU execution.

Default: `FCUTPN`

Example

The following command inserts the PMBIST FCU logic as described in the configuration file, by default, to the design module at the highest level of hierarchy of the design.

```
insert_dft hard_repair -config_file ../hri_inputs/my_hri_configuration.txt
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these constraints:	define_dft dft_configuration_mode on page 925
	define_dft mbist_clock on page 944
	define_dft pmbist_direct_access on page 958
	define_dft test_mode on page 995
Affects these commands:	insert_dft boundary_scan on page 1026
	insert_dft jtag_macro on page 1046
Related commands:	insert_dft pmbist on page 1065
	read_memory_view on page 1117
	read_pmbist_interface_files on page 1120
	write_pmbist_interface_files on page 1207
	write_pmbist_testbench on page 1210
Affected by this attribute:	pmbist_use
Related attributes:	mbist_enable_shared_library_domain_set
	pmbist_enable_multiple_views
	pmbist_instruction_set
	Memory Data Bit Structure Attributes
	Memory Libcell Attributes
	Memory Libpin Action Attributes
	Memory Libpin Alias Attributes
	Memory Spare Column Attributes
	Memory Spare Column Map Address Attributes
	Memory Spare Column Map Data Attributes
	Memory Spare Row Attributes
	Memory Spare Row Map Address Attributes
	Write Mask Bit Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft jtag_macro

```
insert_dft jtag_macro [-boundary_type {IEEE_11491|IEEE_11496}]
     [-dont_map] [-jtag_macro_location instance]
     [-generate_wrck]
     [-insert_without_pad_logic [-create_ports]
       [-dont_create_DFT_TDO_enable_port]]
     [-preserve_tdo_connection]
     [-tck port] [-tdi port] [-tdo port] [-tms port]
     [-trst port] [-no_trst_test_signal]
     [-power_on_reset pin|port] [-design design]
```

Inserts a JTAG Macro (if it is not already instantiated in the design).

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-boundary_type {IEEE_11491 | IEEE_11496}`

Specifies the boundary scan architecture to use for the inserted JTAG Macro.

Default: `IEEE_11491`

`-create_ports`

Specifies whether to create the TAP ports if they do not exist.

If you do not specify already existing TAP signals using the `-tdi`, `-tdo`, `-tms`, `-trst`, and `-tck` options, the missing ports are created and named as *prefix*`tdi`, *prefix*`tdo`, *prefix*`tms`, *prefix*`trst`, and *prefix*`tck`, where *prefix* is the value of the `dft_prefix` root attribute.

Note: Use this option with the `-insert_without_pad_logic` option.

`-design` *design*

Specifies the name of the design in which you want to insert the JTAG Macro. This option is required if you have loaded multiple designs.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dont_create_DFT_TDO_enable_port`

Prevents the creation of the block-level *prefix*`TDO_ENABLE` port which controls the tristate enable pin of the top-level TDO pad.

As a result, the `JTAG_MODULE/JTAG_ENABLE_TDO` output pin will be left unconnected in the netlist. You will need to connect this internal output pin to the appropriate logic to control the three-state enable pin of the JTAG TDO pad to pass boundary scan verification.

Note: Use this option with the `-insert_without_pad_logic` option.

`-dont_map`

Prevents the inserted JTAG Macro from being mapped to technology gates even if the design is already mapped.

`-generate_wrck`

Creates a `JTAG_WRCK` output pin on the JTAG macro. This output port gates the test clock of the JTAG macro (TCK) for all JTAG FSM states other than `Run-Test-Idle`, `CAPTUREDR`, `SHIFTDR`, or `UPDATEDR`.

The `JTAG_WRCK` pin can be used to facilitate interfacing an IEEE 1500 wrapper serial control (WSC) bus to a JTAG macro.

`-insert_without_pad_logic`

Specifies whether to insert the JTAG Macro into a design that does not have pad logic.

Unless you specify the `-dont_create_DFT_TDO_enable_port` option, an additional port, *prefix*`TDO_ENABLE` will be created for the enable signal used to control the tristate pin of the top-level TDO pad.

`-jtag_macro_location` *instance*

Specifies a hierarchical instance in which to insert the JTAG Macro.

`-no_trst_test_signal`

Prevents defining an active-low test signal on the TRST port and causes a formal verification constraint with a value of ‘0’ to be added for the TRST port for the revised design.

`-power_on_reset {`*pin* | *port*`}`

Specifies the power-on-reset pin name.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-preserve_tdo_connection`

Preserves the existing net connection to from-core and tristate enable pins of the TDO pad cell.

If you do not specify this option, the existing net connections will be broken and new net connections will be made from the JTAG Macro `JTAG_TDO` and `JTAG_ENABLE_TDO` pins to the from-core and tristate enable pins of the TDO pad cell, respectively.

Note: If you preserve the TDO connections, such that the net is driven by user logic other than the JTAG Macro, the `JTAG MacroJTAG_TDO` pin will be left unconnected in the netlist.

`-tck` *`port`*

Specifies the port name of the driver for the test clock of the JTAG Macro. Specify this option when the existing JTAG port does not use the standard name.

Default: `TCK`

`-tdi` *`port`*

Specifies the port name of the driver for the test data (scan) input of the JTAG Macro. Specify this option when the existing JTAG port does not use the standard name.

Default: `TDI`

`-tdo` *`port`*

Specifies the port name of the test data (scan) output of the JTAG Macro. Specify this option when the existing JTAG port does not use the standard name.

Note: An existing TDO port must have a tristate I/O pad.

Default: `TDO`

`-tms` *`port`*

Specifies the port name of the test mode select input of the JTAG Macro. Specify this option when the existing JTAG port does not use the standard name.

Default: `TMS`

`-trst` *`port`*

Specifies the port name of the (asynchronous) test reset of the JTAG Macro. Specify this option when the existing JTAG port does not use the standard name.

Default: `TRST`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following example inserts the JTAG Macro into the top-level netlist, creates TAP ports on the top-level netlist, and preserves the existing net connection to from-core and tristate enable pins of the TDO pad cell.

```
insert_dft jtag_macro -create_tap_ports design=design_name \
 -preseve_tdo_connection -insert_without_pad_logic
```

- The following example inserts the JTAG Macro into a specified hierarchical instance and connects it to TAP ports without pad logic.

```
insert_dft jtag_macro -inside instance_level design=design_name \
 -insert_without_pad_logic
```

Related Information

Related constraints:	define_dft jtag_instruction on page 933
	define_dft jtag_instruction_register on page 937
Affects these commands:	compress_scan_chains on page 882
	insert_dft pmbist on page 1065
	insert_dft ptam on page 1071
Sets these attributes:	JTAG_PORT Attributes
	dft_jtag_macro_exists

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft lockup_element

```
insert_dft lockup_element
     actual_scan_chain [actual_scan_chain]...
     [-terminal_lockup]
```

Inserts a lockup element as needed in the specified analyzed scan chains.

Analyzed scan chains are chains whose connectivity is traced by the Genus-DFT engine when you define them using the `-analyze` option of the `define_dft scan_chain` command.

Use this command on mapped netlists whose existing (configured) scan chains were either created by hand or using a third-party DFT insertion tool.

Tip

The Genus-DFT engine determines the type of lockup element to be inserted from the value of the `dft_lockup_element_type` design attribute.

Options and Arguments

actual_scan_chain	Specifies the name of an analyzed scan chain.
`-terminal_lockup`	Allows to add a terminal lockup element to the specified analyzed scan chains.

Example

The following example inserts lockup elements as needed in all analyzed scan chains.

```
insert_dft lockup_element [filter analyzed true \
[find /designs/design/dft -actual_scan_chain *]]
```

Related Information

Affected by this attribute: dft_lockup_element_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft logic_bist

```
insert_dft logic_bist
     -bist_clock port [-bist_clock_hookup pin]
     [-lbist_clock_osc_period integer]
     [-lbist_clock_ref_period  integer]
     [ -scan_clock port -test_control test_signal
       [-scan_clock_hookup pin]
     | -dont_scan_lbist_flops]
     [-inside instance]
     [-tester_available port...]
     [-free_running_clocks test_clock...]
     [-control_inputs_0 port...] [-control_inputs_1 port...]
     [-direct_access] [-programmable_direct_access]
     [-programmable_direct_access_parallel_load_enable port]
     [-direct_reset port] [-direct_logic_bist_enable port]
     [-direct_reset_negedge] [-direct_bist_done_output port]
     [-direct_bist_pass_output port] [-direct_bist_fail_output port]
     [-scan_patterns integer] [-scan_pattern_counter_length integer]
       [-scan_pattern_counter_parallel_load_pins {pin|port}]]
     [-set_patterns integer][-set_pattern_counter_length integer]
       [-set_pattern_counter_parallel_load_pins {pin|port}]]
     [-async_set_signals test_signal]
     [-reset_patterns integer] [-reset_pattern_counter_length integer]
       [-reset_pattern_counter_parallel_load_pins {pin|port}]]
     [-async_reset_signals test_signal]
     [-static_patterns integer] [-static_pattern_counter_length integer]
       [-static_pattern_counter_parallel_load_pins {pin|port}]]
     [-dynamic_patterns integer] [-dynamic_pattern_counter_length integer]
       [-dynamic_pattern_counter_parallel_load_pins {pin|port}]]
     [-scan_channel_counter_length integer]
       [-scan_channel_counter_parallel_load_pins {pin|port}]]
     [-scan_window integer] [-scan_window_counter_length integer]
       [-scan_window_counter_parallel_load_pins {pin|port}]]
     [-scan_window_pulse_value integer] [-scan_enable_delay integer]
     [-scan_enable_delay_counter_length integer]
       [-scan_enable_delay_counter_parallel_load_pins {pin|port}]]
     [-capture_window integer] [-capture_window_counter_length integer
       [-capture_window_counter_parallel_load_pins {pin|port|hpin}...]]
     [-capture_window_capture_value integer]
     [-capture_window_launch_value integer]
     [-set_reset_test_window integer] [-set_reset_pulse_width integer]
     [-capture_clock_stagger_ratio integer]
     [-scan_clock_stagger_ratio integer] [-add_masking]
     [-parallel_load_mask [-mask_parallel_load_pins {pin|port}]]]
     [-parallel_load_misr [-misr_parallel_load_pins {pin|port}]
       [-misr_parallel_unload_pins {pin|port}]]
     [-parallel_load_control_bits [-control_bits_parallel_load_pins {pin|port}...]
     [-parallel_load_prgp [-prgp_parallel_load_pins {pin|port}]]
     [-auto_create_programmable_direct_access_parallel_ports]
     [-dont_use_opcg] [-preview] [design]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Inserts Logic Bist (LBIST) logic into the design.

When the length of any LBIST counter is explicitly specified using the corresponding `*_counter_length` option, that counter will be programmable in `SETBIST` and programmable direct-access LBIST. Otherwise it will always be internally initialized to its default value, same as for `RUNBIST` and (non-programmable) direct-access.

If internal pins are used to drive the programmable direct-access LBIST parallel load interface or the MISR parallel unload interface is either left unconnected or is connected to internal pins, the LBIST simulation environment that is generated by default will not work. You will need to use your own simulation environment where both LBIST as well as the proprietary parallel interface used to operate it, must be functional

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-add_masking`
: Inserts the LBIST macro including the optional channel masking logic (only available with SETBIST support or the programmable direct-access LBIST support).

`-async_set_signals` *`test_signal`*
: Specifies the list of asynchronous set signals that are controlled during set testing.

`-async_reset_signals` *`test_signal`*
: Specifies the list of asynchronous reset signals that are controlled during reset testing.

`-auto_create_programmable_direct_access_parallel_ports`
: Creates ports to connect to all pins of the programmable direct-access parallel load and unload interfaces of the LBIST macro for which no pins have been specified through the corresponding option.

`-bist_clock` *`port`*
: Specifies the port to which the LBIST oscillator clock is applied. You can specify either a top-level dedicated port or an oscillator reference clock.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-bist_clock_hookup` *pin*

Specifies the hookup pin for the LBIST oscillator clock.

Usually the PLL output is used as hookup for the LBIST oscillator clock.

`-capture_clock_stagger_ratio` *integer*

Controls staggering of the capture clocks. The specified value determines the number of bistck pulses after which the capture clock is staggered. Specify 0, 1, 2, 4 or 8. If you specify 0, the capture clocks will not be staggered.

`-capture_window` *integer*

Specifies the default value of the capture window.

`-capture_window_capture_value` *integer*

Specifies the value of the capture window (down) counter at which the (first) capture clock pulse is issued.

`-capture_window_counter_length` *integer*

Specifies the length of the capture window counter that is part of the LBIST macro.

`-capture_window_counter_parallel_load_pins {`*pin*`|`*port*`}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the capture window counter.

This option must be specified together with the `-capture_window_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-capture_window_launch_value` *integer*

Specifies the value of the capture window (down) counter at which the launch clock pulse is issued.

`-control_bits_parallel_load_pins {`*pin*`|`*port*`}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the LBIST control bits.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This option must be specified together with the `-parallel_load_control_bits` option.

If omitted, constants are used to initialize the LBIST control bits to their default values, unless port auto-creation is also selected.

`-control_inputs_0` *`port_list`*

Specifies the list of ports that must be controlled to a logic zero value during LBIST mode.

Use this option if no boundary scan cell or wrapper cell is inserted on these ports.

`-control_inputs_1` *`port_list`*

Specifies the list of ports that must be controlled to a logic one value during LBIST mode.

Use this option if no boundary scan cell or wrapper cell is inserted on these ports.

`design`

Inserts the LBIST logic in the specified top-level design.

`-direct_access`

Specifies whether to insert LBIST without the support from the JTAG Macro.

`-direct_bist_done_output` *`port`*

Specifies the top-level port to use to monitor LBIST completion.

`-direct_bist_fail_output` *`port`*

Specifies the top-level port used to monitor if LBIST fails.

`-direct_bist_pass_output` *`port`*

Specifies the top-level port used to monitor if LBIST passes.

`-direct_logic_bist_enable` *`port`*

Specifies a dedicated top-level port used to enable the direct-access LBIST. This option is required for the direct access support.

`-direct_reset` *`port`*

Specifies a dedicated top-level port used to reset the direct-access LBIST. This option is required for the direct access support.

`-direct_reset_negedge`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies whether the reset signal for the direct-access LBIST is active low.

`-dont_scan_lbist_flops`

Inserts the LBIST macro without converting its flops to scan flops, connecting them in scan chains and prepending them to the full scan chains.

`-dont_use_opcg` Inserts the LBIST macro without using the inserted OPCG logic.

`-dynamic_pattern_counter_length` *`integer`*

Specifies the length of the dynamic pattern counter that is part of the LBIST macro.

`-dynamic_pattern_counter_parallel_load_pins {`*`pin`*`|`*`port`*`}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the dynamic pattern counter.

This option must be specified together with the `-dynamic_pattern_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-dynamic_patterns` *`integer`*

Specifies the default number of dynamic set test patterns to be executed.

`-free_running_clocks` *`test_clock_list`*

Specifies the free running test clocks in the design. These clocks are intercepted in LBIST mode by special mux structures to prevent clock glitches.

`-inside` *`instance`* Specifies the instance in which to instantiate the LBIST logic.

`-lbist_clock_osc_period` *`integer`*

Specifies the period of the LBIST oscillator clock in ps.

`-lbist_clock_ref_period` *`integer`*

Specifies the period of the LBIST reference clock in ps.

By default, the LBIST logic is inserted as a hierarchical instance in the top-level of the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-mask_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the mask.

This option can only be specified together with the `-add_masking` and `-parallel_load_mask` options.

By default, constants will be used to initialize the mask to its default value, unless port auto-creation is also selected.

`-misr_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the MISR.

This option must be specified together with the `-parallel_load_misr` option.

By default, constants will be used to initialize the MISR to its default value, unless port auto-creation is also selected.

`-misr_parallel_unload_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel unload interface of the LBIST macro to observe the MISR signature.

By default, the MISR outputs will be left unconnected for the user to connect, unless port auto-creation is also selected.

`-parallel_load_control_bits`

Inserts programmable direct-access LBIST with a parallel load interface for the LBIST control bits.

By default, the LBIST control bits will be internally initialized to their default values.

`-parallel_load_mask`

Inserts programmable direct-access LBIST with a parallel load interface for the mask.

Specify this option with the `-add_masking` option.

By default, the mask will be internally initialized to its default value (all 1's) which allows all channels to be observed by the MISR.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-parallel_load_misr`

Inserts programmable direct-access LBIST with a parallel load interface for the MISR.

By default, the MISR will be internally initialized to its default value (all 0's).

`-parallel_load_prpg`

Inserts programmable direct-access LBIST with a parallel load interface for the PRPG.

If omitted, the PRPG will be internally initialized to its default value (all 1's).

`-preview` Shows the potential changes without making any modifications to the netlist.

`-programmable_direct_access`

Provides programmable direct access support.

`-programmable_direct_access_parallel_load_enable`

Specifies the programmable direct access parallel load enable port.

`-prpg_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the PRPG.

This option can only be specified together with option '-parallel_load_prpg'.

By default, constants will be used to initialize the PRPG to its default value, unless port auto-creation is also selected.

`-reset_pattern_counter_length integer`

Specifies the length of the reset pattern counter that is part of the LBIST macro.

`-reset_pattern_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the reset pattern counter.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

This option must be specified together with the `-reset_pattern_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-reset_patterns` *integer*

Specifies the default number of reset test patterns to be executed.

`-scan_channel_counter_length` *integer*

Specifies the length of the scan channel counter that is part of the LBIST macro.

`-scan_channel_counter_parallel_load_pins {`*pin*`|`*port*`}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the scan channel counter.

This option must be specified together with the `-scan_channel_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-scan_clock` *port* Specifies the port to which the test clock is applied that drives the LBIST flops when they are prepended to full scan chains.

`-scan_clock_hookup` *pin*

Specifies the hookup pin for the test clock that drives the LBIST flops when they are prepended to the full scan chains.

`-scan_clock_stagger_ratio` *integer*

Controls staggering of the scan clocks. The specified value determines the number of bistck pulses after which the scan clock is staggered. Specify 0, 1, 2, 4 or 8. If you specify 0, the scan clocks will not be staggered.

`-scan_enable_delay` *integer*

Specifies the default delay after the scan enable signal toggles.

`-scan_enable_delay_counter_length` *integer*

Specifies the length of the scan enable delay counter that is part of the LBIST macro.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-scan_enable_delay_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the scan enable delay counter.

This option must be specified together with the `-scan_enable_delay_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-scan_pattern_counter_length integer`

Specifies the length of the scan pattern counter that is part of the LBIST macro.

`-scan_pattern_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the scan pattern counter.

This option must be specified together with the `-scan_pattern_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-scan_patterns integer`

Specifies the default number of scan test patterns to be executed. The specified integer must be a multiple of 64.

`-scan_window integer`

Specifies the default value of the scan window.

`-scan_window_counter_length integer`

Specifies the length of the scan window counter that is part of the LBIST macro.

`-scan_window_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the scan window counter.

This option must be specified together with the `-scan_window_counter_length` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-scan_window_pulse_value` *integer*

Specifies the value of the scan window (down) counter at which the (first) scan clock pulse is issued.

`-set_reset_pulse_width` *integer*

Specifies the width of the asynchronous set/reset pulse for the set/reset tests.

`-set_reset_test_window` *integer*

Specifies the value of the test window for the asynchronous set/reset tests.

`-scan_window` *integer*

Specifies the default value of the scan window.

`-scan_window_counter_length` *integer*

Specifies the length of the scan window counter that is part of the LBIST macro.

`-scan_window_counter_parallel_load_pins {`*pin*`|`*port*`}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the scan window counter.

This option must be specified together with the `-scan_window_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-scan_window_pulse_value` *integer*

Specifies the value of the scan window (down) counter at which the (first) scan clock pulse is issued.

`-set_pattern_counter_length` *integer*

Specifies the length of the set pattern counter that is part of the LBIST macro.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-set_pattern_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the set pattern counter.

This option must be specified together with the `-set_pattern_counter_length` option.

If omitted, constants are used to initialize LBIST to the specified default value, unless port auto-creation is also selected.

`-set_patterns integer`

Specifies the default number of set test patterns to be executed.

`-set_reset_pulse_width integer`

Specifies the width of the asynchronous set/reset pulse for the set/reset tests.

`-set_reset_test_window integer`

Specifies the value of the test window for the asynchronous set/reset tests.

`-stagger_clocks` Inserts the LBIST macro with scan/capture clock staggering support.

`-static_pattern_counter_length integer`

Specifies the length of the static pattern counter that is part of the LBIST macro.

`-static_pattern_counter_parallel_load_pins {pin|port}`

Specifies the ports or pins (from MSB to LSB) to be connected to the programmable direct-access parallel load interface of the LBIST macro to initialize the static pattern counter.

This option must be specified together with the `-static_pattern_counter_length` option.

`-static_patterns integer`

Specifies the default number of static set test patterns to be executed.

`-test_control test_signal`

Specifies the test signal that, when active, brings the LBIST logic into its ATPG test mode in order to test it.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-tester_available` *`port`*

Specifies the ports that are available on the tester and that therefore can be skipped for test point insertion. These ports are written out to the pinassign file if defined as test signal and test clocks.

Related Information

Affected by these constraints:	define_dft test_mode on page 995
Related commands:	write_dft_deterministic_test_points on page 1169
	write_logic_bist_macro on page 1201

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft opcg

```
insert_dft opcg
     -opcg_load_clock test_clock
     -scan_clock test_clock
     -test_enable test_signal
     [-opcg_enable test_signal]
     [-preview] [design]
```

Inserts `OPCG` logic that generates on-chip launch and capture clocks for testing of at-speed delay defects.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-design` *design*

Specifies the name of the design for which to insert OPCG logic.

This option is only required if multiple designs are loaded.

If there is only one top-level design, you can omit the design name. In this case, the tool will use the top-level design of the design hierarchy.

`-opcg_enable` *test_signal*

Specifies the test signal to be used to enable the `OPCG` mode.

The test signal must have been previously defined by a `define_dft test_mode` constraint.

`-opcg_load_clock` *test_clock*

Specifies test clock that will be used to clock the side scan chains (scan chains in the `OPCG` logic).

The test signal must have been previously defined by a `define_dft test_clock` constraint.

`-preview`

Shows the potential changes for the `OPCG` logic, without making any modifications to the netlist.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-scan_clock` *`test_clock`*

Specifies the test clock that must be used for shifting the `OPCG` logic in fullscan mode.

The test signal must have been previously defined by a `define_dft test_clock` constraint.

`-test_enable` *`test_signal`*

Specifies the test signal to be used to gate the `OPCG` enable.

The test signal must have been previously defined by a `define_dft test_mode` constraint.

Example

```
insert_dft opcg -opcg_enable OPCGMODE  -opcg_load_clock opcg_load_clk \
-scan_clock scanclk -test_enable testmode
```

Related Information

Related constraints:

define_dft shift_enable on page 978

define_dft test_clock on page 990

define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft pmbist

```
insert_dft pmbist
     [-config_file config_file | -preview]
     [ -connect_to_jtag | -dont_create_pmbist_ports
     | -direct_access_only [-dont_create_pmbist_ports]]
     [-dont_scan] [-module_prefix string]
     [-dft_configuration_mode mode]
     [-alt_dft_configuration_mode mode]
     [-allow_unused_instructions]
     [-schedule_instruction string]
     [-check_instruction string]
     [-testplan_instruction string]
     [-constraint_instruction string]
     [-rom_instruction string]
     [-diagnostics_instruction string]
     [-rom_diagnostics_instruction string]
     [-redundancy_instruction string]
     [-rompath string] [-romcontentsfile string]
     [-amu_location {design|subdesign|instance}]
     [-directory directory_path] [design]
```

Inserts Programmable Memory Built-In-Self-Test (PMBIST) logic in the design for memories based on the definitions in the configuration file specified using the `-config_file` option.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-allow_unused_instructions`

Enables the creation of all possible `jtag_instruciton_decode*` ports for PMBIST instructions for the block being processed by this command. This option assures a common interface on the block (independent of the functions enabled within this block).

Default: pmbist

`-amu_location {design|subdesign|instance}`

Specifies the location for the algorithm memory unit which includes the programmable algorithm information.

Default: top-level design

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-check_instruction` *`string`*

Specifies the `MBISTCHK` instruction that must be loaded into the IEEE 1149.x TAP controller to check the results of PMBIST execution.

Default: `MBISTCHK`

`-config_file` *`config_file`*

Specifies the file that contains the user-defined configuration parameters which selects the PMBIST features and controls for the insertion of the test logic for targeted memories.

`-connect_to_jtag`

Connects PMBIST logic's JTAG interface pins to a pre-instantiated JTAG macro (either Cadence's or Third party).

Use this option when the design is the top level of the chip, and you have instantiated the JTAG macro and will use it to access the PMBIST engines.

`-constraint_instruction` *`string`*

Specifies the `MBISTAMR` instruction that must be loaded into the IEEE 1149.x TAP controller to access programmable algorithm memory and test conditions.

Default: `MBISTAMR`

`design`

Specifies the name of a top-level design.

`-dft_configuration_mode` *`mode`*

Specifies the configuration mode in which PMBIST will be operating. PMBIST test pin verification must be done in this mode as well.

Default: `pmbist`

`-diagnostics_instruction` *`string`*

Specifies the JTAG instruction that must be loaded into the IEEE 1149.x TAP controller when the design contains any memory that needs to be tested using the diagnostic isolation options.

Default: `MBISTDIAG`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-direct_access_only` Specifies whether to insert PMBIST without the support from the JTAG Macro. The memory testing is controlled and examined directly from the PMBIST direct access ports/pins without using the JTAG macro. In this case, any information for the JTAG macro is ignored even when a JTAG macro is specified in the design.

`-directory` *`directory_path`*

Specifies the directory in which PMBIST must create all intermediate files for its internal use.

Default: *`current_working_directory/`*`pmbist_`*`design`*

`-dont_create_pmbist_ports`

Prevents creation of PMBIST control ports for subsequent JTAG macro and direct access connection at the top-level instance.

Use this option when the instance is the top level of the chip, and you have not yet instantiated the JTAG macro, or you use the direct access method and these ports were predefined.

`-dont_scan` Allows the insertion of non-scannable PMBIST logic.

Use this option if ATPG is not required for the PMBIST logic or for the design. This implies that test signals do not need to be specified for the PMBIST logic.

`-module_prefix` *`string`*

Specifies an additional character string to append to the default prefix `tem`. The string is placed on all modules created and inserted by the `insert_dft pmbist` command.

Note: This option is required in case of a PMBIST block-level insertion flow.

`-preview` Requests the creation of a configuration file template without performing insertion. The template file can be used to review configuration file content or as a basis to further edit content.

`-redundancy_instruction` *`string`*

Specifies the JTAG instruction that must be loaded into the IEEE 1149.x TAP controller for redundancy analysis.

Default: `MBISTRAR`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-rom_diagnostics_instruction` *`string`*

Specifies the JTAG instruction that must be loaded into the IEEE 1149.x TAP controller when the design contains ROMs that need to be tested using the predefined diagnostic test plan

Default: `MBISTROMDIAG`

`-rom_instruction` *`string`*

Specifies the JTAG instruction that must be loaded into the IEEE 1149.x TAP controller when the design contains ROMs that need to be tested using programmable test plans.

Default: `MBISTROM`

`-romcontentsfile` *`string`*

Specifies the list of ROM data map files located in one of the directories specified with the `-rompath` option.

Note: This option is especially important when hardwired testplans are used for ROM testing as the signature is hardwired into the PMBIST logic.

`-rompath` *`string`*

Specifies a list of directory paths where the ROM data map files reside.

Note: This option is especially important when hardwired testplans are used for ROM testing as the signature is hardwired into the PMBIST logic.

`-schedule_instruction` *`string`*

Specifies the `MBISTSCH` instruction that must be loaded into the IEEE 1149.x TAP controller to schedule targeted memories for PMBIST execution.

Default: `MBISTSCH`

`-testplan_instruction` *`string`*

Specifies the `MBISTTPN` instruction that must be loaded into the IEEE 1149.x TAP controller for test plan selection during PMBIST execution.

Default: `MBISTTPN`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command analyzes the netlist to identify memory instances and generates a PMBIST configuration file template for this design.

```
insert_dft pmbist  -preview
```

- The following command inserts the PMBIST logic as described in the configuration file, by default, to the design module at the highest level of hierarchy of the design.

```
insert_dft pmbist -config_file ../et_inputs/my_configuration.txt
```

Related Information

Affected by these constraints:	define_dft dft_configuration_mode on page 925
	define_dft mbist_clock on page 944
	define_dft pmbist_direct_access on page 958
	define_dft test_mode on page 995
Affects these commands:	insert_dft boundary_scan on page 1026
	insert_dft jtag_macro on page 1046
Related commands:	read_memory_view on page 1117
	read_pmbist_interface_files on page 1120
	write_pmbist_interface_files on page 1207
	write_pmbist_testbench on page 1210
Affected by this attribute:	pmbist_use
Related attributes:	mbist_enable_shared_library_domain_set
	pmbist_enable_multiple_views
	pmbist_instruction_set
	Memory Data Bit Structure Attributes
	Memory Libcell Attributes
	Memory Libpin Action Attributes
	Memory Libpin Alias Attributes
	Memory Spare Column Attributes
	Memory Spare Column Map Address Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Memory Spare Column Map Data Attributes
Memory Spare Row Attributes
Memory Spare Row Map Address Attributes
Write Mask Bit Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft ptam

```
insert_dft ptam
     -instruction string
     -power_test_enable {pin|port}
     [-power_test_enable_active {low|high} ]
     [-shift_enable test_signal ]
     [-connect_to_jtag]
     [-directory string ] [-preview]
     [-dont_map ] [-dont_check_dft_rules ]
```

Inserts Power Test Access Mechanism (PTAM) logic to facilitate chip power management during test.

Note: Some files may require customization according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-connect_to_jtag`

Connects the PTAM logic to the JTAG macro instance. The JTAG macro instance must first be created using either the `insert_dft boundary_scan` command or the `insert_dft jtag_macro` command. If `-connect_to_jtag` is not specified and a JTAG macro instance is detected in the current session, a warning message will be issued to notify of the JTAG macro instance's existence, otherwise there will be no attempt to connect to a JTAG macro instance.

`-directory` *`string`*

Specifies the directory to which the mode initialization and pin assign files are written.

Default: *`current_working_directory`*`/ptam`

`-dont_check_dft_rules`

Prevents the DFT rules from being automatically checked after PTAM insertion.

`-dont_map`

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-instruction` *`string`*

Specifies the name of the instruction which must be loaded into the IEEE 1149.x TAP controller instruction register to access the PTAM test data register.

`-power_test_enable {`*`pin`* `|` *`port`*`}`

Identifies the power-test-enable pin or port. Asserting this pin to an active state will enable the PTAM logic to override the design's power manager control pins. This serves as a master mode control signal.

Note: This cannot be the same pin or port identified by `define_dft test_mode` that control pin sharing if the PTAM I/O are shared

`-power_test_enable_active {high |low}`

Specifies the active value for the power-test-enable pin or port.

Default: `high`

`-preview`

Shows the potential changes without making any modifications to the netlist.

`-shift_enable` *`test_signal`*

Designates a shift-enable test signal used to override the PTAM gating logic of the power manager output control signals during scan test.

If you omit this option, the default shift-enable signal specified using `define_dft shift_enable` is used as selected by its `default_shift_enable` attribute.

Examples

- The following command inserts the PTAM logic into design `chip_top`.

```
insert dft ptam -instruction PTAM -power_test_enable /chip_top/PwrTe \
-directory ${workdir}/testmode_data
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands: read_power_intent on page 1285

define_dft shift_enable on page 978

create_isolation_rule in the *Common Power Format Language Reference*

create_power_domain in the *Common Power Format Language Reference*

create_state_retention_rule in the *Common Power Format Language Reference*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft scan_power_gating

```
insert_dft scan_power_gating
     -max_number_of_testpoints integer [-min_slack integer]
     {-test_control test_signal | -preview}
     [-report_virtual_scan_power [-power_mode mode]
          "clockFreq [flopTogglePercent]" -preview]
     [-input_tp_file file]  [-output_tp_file file]
      [-dont_check_dft_rules] [design]
```

Inserts gating logic at selected flop outputs to minimize switching power during scan shift. This command must be run prior to building the actual scan chains in the design.

The usage of this command and the flows that it can be used with, are documented in the section: Gating Functional Paths to Reduce Scan Shift Power in *Genus Design for Test Guide for Legacy UI*.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

design

Specifies the name of the top-level design on which to perform test point insertion.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dont_check_dft_rules`

Prevents the DFT rules from being automatically checked after inserting scan power gating.

`-input_tp_file` *file*

Specifies the name of the file containing the test point locations. The file is specified in Modus format.

If you omit this option, the test point locations are read from:

- The file specified with the `-output_tp_file` option
- The following file in the working directory if you did not specify any file:

  `TB/testresults/TestPointInsertion.ASSUMESCAN.expt.`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max_number_of_testpoints` *`integer`*

Specifies the maximum number of test points to be inserted. Selection is based on the specified number of identified test points and the weight of the point resulting from logic cone analysis.The weight indicates the probability of a higher power level if toggling occurs.

`-min_slack` *`integer`*

Limits the insertion of a test point to those nodes that have the specified minimum slack (in ps).

Default: `2000`

`-output_tp_file` *`file`*

Specifies the name of the generated output file containing the recommended test points.

If you do not specify this option, the test point locations are written to the following file in the working directory:

`TB/testresults/TestPointInsertion.ASSUMESCAN.expt`

`-power_mode` *`mode`*

Specifies the power mode for which the test power must be reported.

Specify this option with the `-report_virtual_scan_power` option when the design has multiple power modes.

`-preview`

Reports the test points to be added, without modifying the design.

`-report_virtual_scan_power "`*`clockFreq`* `[`*`flopTogglePercent`*`]"`

Performs a *virtual* insertion of the test points into the design and measures their impact on the reduction of scan power. Use a floating value to specify the test clock frequency (in MHZ) and the flop-toggle percentage to use in the measurement. If the flop toggle percentage is not specified, it will default to 50% of the test clock frequency.

Note: This option is only valid with the `-preview` option.

`-test_control` *`test_signal`*

Specifies the test signal to enable the test point. This option is not required when the command is run with the `-preview` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode -scan_shift` constraint.

Related Information

Gating Functional Paths to Reduce Scan Shift Power in *Genus Design for Test Guide for Legacy UI.*

Affected by these constraints: define_dft shift_enable on page 978
define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft shadow_logic

```
insert_dft shadow_logic
     {-around instances [-test_control test_signal]
       {-mode bypass
       |-mode no_share -test_clock_pin {port|pin}
          [-rise |-fall]
       |-mode share -test_clock_pin {port|pin}
          [-fall | -rise] }
       [-exclude pins | -only pins ] [-group pins]...
       [-balance]
     |-auto [-minimum_shadow_logic_pins integer]
       [-exclude_shadow_logic instances]
       [-test_control test_signal]
       [-test_clock_pin {port|pin}] }
     [-dont_map] [-preview] [design]
```

This command either

- Automatically inserts shadow logic around logic abstract and timing models (by adding observable flops in non-share mode) when you use the `-auto` option.
- Manually inserts bypass logic and scannable logic with or without register sharing when you use the `-around` option.

  You can insert two basic types of DFT shadow logic around a particular instance: bypass and scannable logic. Each shadow logic flip-flop can implement one control point and one observation point at the same time.

  If you want to share observation and control points, either by setting `-mode` to `share` or `bypass`, the following sharing criteria are observed:

  - If you specify `-group`, the specified inputs and outputs are grouped together as indicated
  - For the remaining inputs and outputs that are not listed with `-group`, the first input will share the flip-flop with or be connected to the first output, the second input with the second output, and so on. The order is that specified in the HDL interface declaration.

Note: Test points are added for unidirectional pins only. Bidirectional pins and pins associated with test clock objects are skipped.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-around` *`instances`*
: Specifies the instances around which the DFT shadow logic must be inserted. Specify a hierarchical instance name.

`-auto`
: Automatically inserts shadow logic around logic abstract and timing models (by adding observable flops in non-share mode).

`-balance`
: Groups unmatched input and output pins to have a balanced number of groups.

`design`
: Specifies the name of the top-level design in which to insert shadow logic.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dont_map`
: Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

`-exclude` *`pins`*
: Prevents the specified pins from being considered for shadow logic insertion.

`-exclude_shadow_logic {`*`instance...`*`}`
: Excludes automatic shadow logic insertion for the specified instances. You can only specify instances of blackboxes or timing models.

`-group` *`pins`*
: Specifies the pins to group when also using `-mode share` or `-mode bypass`. Each group can have multiple input pins and multiple output pins. Format the groups as follows:

  $\{input_i \ldots output_j \ldots\}$

  Separate the pins by spaces. If you have more than one group, you must specify multiple `-group` options.

  Note: If `-mode` is set to `bypass`, each group must have at least one input and one output. Otherwise, the number of inputs or outputs can be equal or larger than zero.

`-minimum_shadow_logic_pins` *`integer`*
: Limits shadow logic insertion to logic abstract or timing models that have more pins (inputs and outputs) than the specified value.

  Default: `10`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-mode`
: Specifies the type of shadow logic to insert.

  `bypass`
  : Implements bypass logic. If you specify this option, you must balance the number of inputs and outputs.

  `no_share`
  : Inserts one scannable observation test point per input and one scannable control test point per output.

  `share`
  : Pairs each input with an output and uses one scannable control and observation test point for each pair. If there are a different number of inputs and outputs, uses one scannable observe (or control) test point is used for each remaining input (or output).

`-only` *`pins`*
: Restricts the pins to be considered for shadow logic insertion to the specified ones.

`-preview`
: Shows the potential changes, without making any modifications to the netlist.

`[-rise|-fall]`
: Specifies the edge of the test clock that is active during test mode operation. These options are only valid in conjunction with `-test_clock_pin`.

  Default: `-rise`

`-test_clock_pin {`*`port`* `|` *`pin`*`}`
: Specifies the test clock that drives the clock pin of the shadow flip-flops. You can specify a port or pin that drives the test clock.

`-test_control` *`test_signal`*
: Specifies the test signal to use to control DFT logic (the multiplexers after the controlling points).

  Note: You must have specified the test signal using either the `define_dft shift_enable` or `define_dft test_mode` constraint.

Examples

In the following examples, the logic before the ATPG-untestable module is not observable and the logic after it is not controllable. Following is the Verilog input code for the ATPG-untestable module and its instantiation:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
module blackbox (i1,i2, o1,o2,o3)
    input i1,i2;
    output o1,o2,o3;
...
blackbox U1 (.i1(n_1), .i2(n_2), .o1(n_3), .o2(n_4), .o3(n_5));
...
```

- Using the following examples, bypass logic is used to make the two inputs observable and the three outputs controllable. The first command pairs input `i1` to output `o1`, and input `i2` to output `o3` (skipping `o2`).The second command pairs input `i1` and output `o2`.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -mode bypass \
    -exclude o2
insert_dft shadow_logic -around U1 -test_control my_TM -mode bypass \
    -only {i1 o2}
```

- The following example uses scannable test points and shares these test points as control and observation points.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock_pin CK \
-mode share
```

- The following example uses scannable test points but does not share these test points for control and observation points.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock CK \
-mode no_share
```

- The following example uses scannable test points, shares these test points for control and observation points, and controls by grouping which pins share a common test point. More specifically, `i1` and `o2` share a test point, and `i2` and `o1`. In addition, no control point is inserted for the net driven by `U1/o3`.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock CK \
-exclude o3 -mode share -group {i1 o2} -group {i2 o1}
```

- The following example automatically inserts shadow logic around all logic abstract and timing model instances.

```
legacy_genus:/> insert_dft shadow_logic -auto -test_control my_tm
INFO: Using test clock pin '/designs/data_ram_fj/ports_in/CK' for testpoint
insertion.
WARNING: pin 'U_BIST_CONTROLLER/I_BIST_WR_ENABLE_REG/CP' is skipped from
shadow DFT insertion since it is driven by a clock

WARNING: pin 'U_BIST_CONTROLLER/I_BIST_WR_ENABLE_REG/QN' is skipped from
shadow DFT insertion because it has no load

Total number of test points inserted: 6
....
WARNING: pin 'U_REAL_DATA_RAM3/CK' is skipped from shadow DFT insertion since
it is driven by a clock
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Total number of test points inserted: 79

  Mapping shadow DFT logic...
...
652
```

Note: The command returns the total number of test points inserted.

Related Information

Affects these commands:	check_dft_rules on page 870
	report dft_registers on page 1128
Related constraints:	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft shift_register_test_points

```
insert_dft shift_register_test_points [-preview]
     [-write_to_log_file file] [-shift_enable test_signal]
     [-max_number_of_test_points integer]
     [-incremental] [-design design]
```

Allows you to add shift register test points in a mapped netlist to identify additional shift registers. You can insert the test points in one of the following ways:

- By automatically identifying the test points and manually selecting the test points.
- By automatically identifying the test points and inserting them in one step.

Options and Arguments

`-design` *design*	Specifies the name of the design in which to insert the test points.
`-incremental`	Identifies shift registers in incremental mode.
`-max_number_of_test_points` *integer*	Specifies the maximum number of test points that can be inserted.
`-preview`	Performs test point identification in preview mode to check the suggested test point locations.
`-shift_enable` *test_signal*	Specifies the controlling shift enable signal.
`-write_to_log_file` *file*	Specifies the file to which to write out the test points. Use this option when you want to manually select the test points.

Related Information

Related constraint:	define_dft shift_enable on page 978

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft test_point

```
insert_dft test_point -location {pin|port}...
     [-test_control test_signal]
     [-gate_clock_test_control test_signal]
     [-gate_clock_for_observe test_signal]
     -type { control_0 | control_1 } |
           {{async_0 | async_1 | async_any | bypass_scan
            | control_node -node {pin|hpin|port}
            | control_observe_0 | control_observe_1
            | control_observe_node -node {pin|hpin|port}
            | control_scan
            | observe_scan [-max_observe_share integer]
            | scan | sync_0 | sync_1 | sync_any }
            -test_clock_pin {pin|hpin|port} [-rise|-fall] }
     [-dont_map]
```

Allows you to manually specify a control or observation test point to be added to the design. Control test points always require the specification of a test-mode signal. Test points that use scannable flip-flops to observe or control a node always require a test-clock signal.

For all of the scannable test points, you need to run `check_dft_rules` after the test point is inserted.

The command returns the path name of the inserted test point when it is a flip-flop.

Important

You can only specify multiple locations when you request to insert a test point of type `observe_scan`. In this case, the tool builds a balanced XOR-tree with all the specified location pins. Additionally, you can control the maximum number of pin locations to be observed by the same observation flop by specifying the `-max_observe_share` option. If a test-control signal is also specified, the tool builds the XOR-tree after each input is AND-ed or OR-ed with the test-control signal. This prevents switching along the XOR-tree when not it test mode. If the test control is active high, gating happens by AND-ing, otherwise by OR-ing. The output of the last XOR-gate is fed to the D input of the observation flip-flop.

Options and Arguments

`-dont_map`
: Prevents the inserted test point from being mapped even if the design is already mapped to the target library.

`-gate_clock_for_observe` *`test_signal`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the test signal to gate the clock for the observe flop.

`-gate_clock_test_control` *`test_signal`*

Specifies the test signal to gate the clock for the control flop.

`-location {`*`port`* `|` *`pin`*`}`

Specifies the location of the control point or observation point. Specify an existing hierarchical pin name or a top-level port. For observation test points, the pin can be an input or output pin. For control test points, the result is different depending on the direction of the location. See the Examples on page 1087.

Note: You can only specify multiple locations for a test point of type `observe_scan`.

Note: If you specify a bidirectional pin, no logic will be inserted unless you specify the direction of the pin.

`-max_observe_share` *`integer`*

Specify the maximum number of locations to be shared for an observe test point.

Note: This option applies only when you set `-type` to `observe_scan`.

`-node {`*`pin`* `|` *`port`*`}`

Specifies the pin or port to insert when `-type` is set to `control_node`, `control_observe_node` or `bypass_scan` and when the signal specified by `-test_control` is active.

`[-rise | -fall]`

Specifies the edge of the specified test clock that is active during test mode operation. These options are only valid in conjunction with `-test_clock`.

Note: You must use the same clock edge when inserting a control flip-flop and an observation flip-flop.

Default: `-rise`

`-test_clock_pin {`*`port`* `|` *`pin`*`}`

Specifies the test clock that drives the clock pin of the inserted flip-flops during test mode operation. You can specify a port or pin that drives the test clock.

This option is required for all type point types set using the `-type` option except those of type `control_0` or `control_1`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-test_control` *`test_signal`*

Specifies the test signal to use to control or observe the specified location point.

Note: You must have specified the test signal using either the `define_shift_enable` or `define_test_mode` constraint.

Note: Test points of type `observe_scan` do not require a test signal.

`-type`

Specifies the type of test point to insert at the specified location when the signal specified by `-test_control` is active. Possible values are:

`async_0`	Inserts an asynchronous control test point that forces the control point to the value 0.
`async_1`	Inserts an asynchronous control test point that forces the control point to the value 1.
`async_any`	Inserts an asynchronous control test point that forces the control point to take either the original value or the inverted value.
`control_0`	Inserts a constant value `0`.
`control_1`	Inserts a constant value `1`.
`control_node`	Inserts an arbitrary node.
`control_observe_0`	Inserts a control and an observation point. The control point is forced to the value 0.
`control_observe_1`	Inserts a control and an observation point. The control point is forced to the value 1.
`control_observe_node`	Inserts a control and an observation point. The control point is forced to the value of the node specified by `-node`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`control_scan`	Inserts a flip-flop to force a particular value at the specified location during test mode operation. The flip-flop must be remapped to a scan flop before connecting it to a scan chain later on. **Note:** This option requires you to specify the `-test_clock_pin` option.
`double_cntrol_0`	Inserts two asynchronous control test points that forces the control point to the value 0.
`double_cntrol_1`	Inserts two asynchronous control test points that forces the control point to the value 1.
`double_cntrol_observe_0`	Inserts an asynchronous control test point that forces the control point to the value 0 and inserts a flip-flop to observe the specified location.
`double_cntrol_observe_1`	Inserts an asynchronous control test point that forces the control point to the value 1 and inserts a flip-flop to observe the specified location.
`observe_scan`	Inserts a flip-flop to observe the specified location. The flip-flop must be remapped to a scan flip-flop before connecting it to a scan chain later on. This option requires you to specify the `-test_clock_pin` option.
`scan`	Inserts a scannable control and observation test point. **Note:** This option requires you to specify the `-test_clock_pin` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`sync_0`	Inserts a synchronous control test point that forces the control point to the value 0.
`sync_1`	Inserts a synchronous control test point that forces the control point to the value 1.
`sync_any`	Inserts a synchronous control test point that forces the control point to take either the original value or the inverted value.

Examples

- The following example inserts a scannable observation test point, using `CLK` to drive:

```
insert_dft test_point -location X/out -test_clock_pin CLK -type observe_scan
```

- The following example inserts a control-1 and scannable observation point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type control_observe_1
```

- The following example inserts a scannable control point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type control_scan
```

- The following example inserts a scannable control and observation test point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type scan
```

- The following example inserts an async control-0 test point at hierarchical pin `X/out`:

```
insert_dft test_point -location X/out -test_control TM -type async_0 \
-test_clock_pin CK -fall.
```

- The following example inserts a synchronous control test point that forces the control point to the value 1 at hierarchical pin `X/out`:

```
insert_dft test_point -location X/out -test_control TM -type sync_1 \
-test_clock_pin CK -fall.
```

- The following example inserts two observation test points, one for `pin1`, `pin2` and `pin3`, and the other for `pin4`, `pin5` and `pin6`.

```
insert_dft test_point -type observe_scan -max_observe_share 3 \
-location pin1 pin2 pin3 pin4 pin5 pin6
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Inserting a Control and Observation Test Point in *Genus Design for Test Guide for Legacy UI*.

Affects these commands:	check_dft_rules on page 870
	connect_scan_chains on page 906
	syn_map on page 529
	syn_opt on page 531
Related constraints:	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995
Related attributes:	Test Clock Attributes
	Test Signal Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft user_test_point

```
insert_dft user_test_point
     -location {pin|port|subport}
     -cell {design|subdesign|libcell}
     {-cfi {pin|port}] | -no_cfi} [-cfo {pin|port}]
      -connect string [-connect string]...
     [-unconnect pin_list] [-name name]
```

Inserts a user-defined test point at the specified location, and hooks it up to the specified pins.

Options and Arguments

`-cell {design|subdesign|libcell}`

Specifies the module or library cell to instantiate. The module can be loaded as a parallel design, or as a subdesign.

`-cfi {pin | port}`

Specifies the cell functional input (CFI) pin or port name (name of the input pin or port that will be automatically connected to the input side of the new broken net of the -location option).

`-cfo {pin | port}`

Specifies the cell functional output (CFO) pin or port name (name of the output pin or port that will be automatically connected to the output side of the new broken net of the -location option).

`-connect string`

Specifies a string consisting of a cell pin and the corresponding source-signal pin to which the cell pin must be connected.

This string has the following format:

`{cell_pin source_pin}`

Use this option to specify most connections to the cell, except for the connections to the CFI and CFO pins.

`-location {pin|port|subport}`

Specifies a pin, port or subport that identifies where the test point must be inserted.

`-name name`

Specifies the instance name to be given to the user-defined test point.

`-no_cfi`

Specifies that the user test point cell has no CFI pin.

`-unconnect pin_list`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies a list of pins on the test point that are intentionally left unconnected. When you specify the unconnected pins with this option, the tool will not issue a warning.

Examples

- The following example inserts design `MyUserTI` in design `top` at the `in1[2]` input port. Port `MyCFI` will be connected to input port `in1[2]`.

```
insert_dft user_test_point -location top/in1[2] -cell /designs/MyUserTI \
-cfi MyCFI -cfo MyCFO -connect {MyShiftEn se} -connect {MyWRCK wck}
```

- Assume pins `D`, `S1` and `C` of the user test point are not connected.
  - If you specify the command without the `-unconnect` option, the tool will issue warnings:

```
legacy_genus:/> insert_dft user_test_point -cell MX4X1 -location en \
-cfi A -cfo Y -connect {B se} -connect {S0 tm}
```

```
Warning : The testpoint pin is not connected. [DFT-296]
        : Pin 'D' of testpoint will be left unconnected.
       : Use 'connect' option to connect the pin or use 'unconnect' option
to suppress the warning message.
Warning : The testpoint pin is not connected. [DFT-296]
        : Pin 'S1' of testpoint will be left unconnected.
Warning : The testpoint pin is not connected. [DFT-296]
        : Pin 'C' of testpoint will be left unconnected.
```

  - If you specify the command with the `-unconnect` option, the tool will only issue a warning for any unconnected pin, not mentioned in the pinlist of the `-unconnect` option:

```
legacy_genus:/> insert_dft user_test_point -cell MX4X1 -location en \
-cfi A -cfo Y -connect {B se} -connect {S0 tm} -unconnect {D S1}
```

```
Warning : The testpoint pin is not connected. [DFT-296]
        : Pin 'C' of testpoint will be left unconnected.
        : Use 'connect' option to connect the pin or use 'unconnect' option
to suppress the warning message.
```

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*:

- Inserting a User-Defined Control and Observation Test Point
- Inserting Scannable Test Points in Existing Scan Chains

Affects these commands: check_dft_rules on page 870

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect_scan_chains on page 906

syn_map on page 529

syn_opt on page 531

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft wir_signal_bits

```
insert_dft wir_signal_bits
     [-create_wrapper_shift_enables_for_delay_test test_signal]
     [-create_cgic_enable_for_wrap test_signal]
     [-preview] [-design design]
```

Inserts a wrapper instruction register or WIR, the first time you invoke the command, and adds the WIR signals defined thus far by the `define_dft test_bus_port -wir_signal` command. Each time you invoke this command, it adds the WIR signals that you defined since the last time you invoked this command.

In addition, the command connects the WIR to the ports corresponding to the Wrapper Serial Control (WSC) signals and Wrapper Serial Input (WSI). The WSC and WSI signals are defined with the `define_dft test_bus_port` command with specific function values.

The command is used when building an IEEE 1500 wrapper around a design block. The usage of this command is documented in *Genus Design for Test Guide for Legacy UI* in the section titled Inserting Wrapper Instruction Register and Defining WIR Signal Bits in chapter Inserting the Wrapper Cells

Options and Arguments

`-create_cgic_enable_for_wrap` *`test_signal`*

Specifies the test signal to be gated by the inverse of the decoded `EXTEST` signal. You can specify this gated signal (`DFTCGENI`) as the `lp_clock_gating_test_signal` to control the test pin of the functional clock gates which feed the core flops. After inserting the core wrapper logic, the test pins of the functional clock gates which feed the shared wrapper cells can be controlled by the `DFTWSEO` signal by rewiring the test pin connections.

`-create_wrapper_shift_enables_for_delay_test` *`test_signal`*

Specifies the shift-enable signal to use to create two new shift-enable signals: the wrapper shift-enable signal used for `INTEST` mode (gated by the decoded `INTEST` signal), and the wrapper shift-enable signal used for `EXTEST` mode (gated by the decoded `EXTEST` signal).

`-design` *`design`*

Specifies the design in which you want to insert the wrapper instruction register.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	This option is required if multiple designs are loaded.
`-preview`	Shows a preview of the WIR signals that would be added to the WIR along with their bit index in the WIR.

Related Information

Inserting the Wrapper Cells in *Genus Design for Test Guide for Legacy UI*

Related command: define_dft test_bus_port on page 985

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft wrapper_cell

```
insert_dft wrapper_cell -location pin_list
     [-floating_location_ok]
     [-dont_wrap_floating_locations_for_hierarchical_pins]
     [-skipped_locations_variable Tcl_variable]
     [-shared_through {buffer|inverters_and_buffers|combinational}
       [-no_mux_before_shared_cell]]
     [-wck pin] -wsen_in pin -wsen_out pin
     {-input_select_cfi {pin|port}
     |-wint {pin|port|constant} }
     {-output_select_cfi {pin|port}
     |-wext {pin|port|constant} [-wcap pin]
     [-guard {0|1} -wig pin -wog pin]
     [-dont_reuse_input_wrappers_for_output_ports] [-exclude_string string]
     [-exclude pin_list] [-exclude_comb_feedthrough_paths]
     [-input_shared_threshold integer]
     [-output_shared_threshold integer]
     [-override_threshold_use_dedicated {pin|port}...]
     [-override_threshold_use_shared {pin|port}...]
     [-override_threshold_use_dedicated_string string]
     [-override_threshold_use_shared_string string]
     [-respect_dft_constants {test_setup | tied}]
     [-wrap_tied_constant_ports] [-inside_core]
     [-map_to_mux_for_dedicated_wrapper]
     [-map_to_mux_for_shared_wrapper_cell]
     [-use_power_domain_of_ports_for_dedicated]
     [-wsen_reuse_or_for_shared_wrapper]
     [-add_xor_feedback]
     [-wint_in_out {pin|hpin|port|constant}]
     [-name segment_prefix] [-preview] [-design design]
```

Selects a built-in IEEE 1500 standard wrapper cell based on the given specifications, inserts it at the specified location, and hooks it up to the specified control signals.

The wrapper cell logic is automatically identified as preserved wrapper-cell segments. You can use these segments in other segments (within nested segments) or specify the segments as elements when building the scan chains.

The command returns the directory path to the `scan_segment` objects that it creates. If multiple locations are specified, the command returns multiple segments. You can find the objects created by the `insert_dft wrapper_cell` in:

`/designs/top_design/dft/scan_segments`

Important

Any segment inserted by this command cannot be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-add_xor_feedback {pin|hpin|port}...`

Specifies to add xor feedback on hold path and use pin or port as an input to xor gate.

`-design design`

Specifies the design in which you want to insert the wrapper cell. This option is required if multiple designs are loaded.

`-dont_reuse_input_wrappers_for_output_ports`

Allows to insert dedicated wrapper cells in the fanin of output ports that are being fed by wrapper cells previously inserted for input ports.

`-dont_wrap_floating_locations_for_hierarchical_pins`

Prevents wrapper cells from being inserted for hierarchical floating locations.

`-exclude pin_list`

Specifies the list of pins or ports to be excluded from wrapper insertion.

`-exclude_comb_feedthrough_paths`

Specifies whether to exclude pins or ports from wrapper insertion if combinational logic is found on the path from input to output.

`-exclude_string string`

Specifies the list of ports to be excluded from wrapper insertion.

`-floating_location_ok`

Specifies to insert a wrapper cell even if the specified pin (location) is floating.

`-guard {0|1}`

Specifies the guard for safe_ value out of a wrapper cell. The safe value prevents testing of one block from interfering with another block.

If this option is not specified, no guard logic will be included in the wrapper cell.

`-input_select_cfi {pin|port}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the source pin name of the decoded control logic for the `select_cfi` signal for input bounding wrapper cells.

Use this option to connect an already decoded signal. Otherwise, control decoder logic may have to be inserted for each cell, and extra control wires will have to be hooked up to each wrapper cell.

`-input_shared_threshold` *`integer`*

Specifies the maximum number of scannable flops that can be shared in wrapper cells. When the number of scannable flops exceeds the specified threshold value, a single dedicated wrapper cell will be inserted for the input port.

Default: `10`

`-inside_core`

Inserts the wrapper cells associated with hierarchical pins inside the core module.

`-location` *`pin_list`*

Specifies one or more pins that identify where the wrapper cell must be inserted.

Note: When specifying the pins of a blackbox instance use the RC pin name to identify the input or output pins.

`-map_to_mux_for_dedicated_wrapper`

Specifies to map the mux logic (associated with the CFI to CFO path and the optional Guard control path in the dedicated wrapper cell) to binary muxes from the target library. The binary muxes will be marked preserved (`preserve` attribute on libcell is set to `true`) for optimization. The instance names given to the muxes inserted for the CFI to CFO path and Guard control path are `dwc_cfi_mux` and `dwc_guard_mux` respectively.

Note: If you omit this option, the technology mapper will select the appropriate cell type(s) which satisfy the timing constraints.

`-map_to_mux_for_shared_wrapper_cell`

Specifies to map the hold mux logic (associated with the CTI path in the shared wrapper cell) to a binary mux from the target library. The binary mux will be marked preserved (`preserve` attribute on libcell is set to `true`) for optimization. The instance name given to the mux is `swc_hold_mux`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: If you omit this option, the technology mapper will select the appropriate cell type(s) which satisfy the timing constraints.

`-name` *`segment_prefix`*

Specifies the segment name prefix.

If you specified a single location pin, and there is no name conflict, the segment name will correspond to the specified prefix, otherwise a unique name will be generated for each segment that is derived from the specified prefix.

`-no_mux_before_shared_cell`

Specifies to prevent the insertion of the hold mux on the test input of shared wrapper cells.

Use this option when the wrapper cells are configured for at-speed testing using delay test. The option applies to those wrapper cell configurations in which the input bounding wrapper cells (and output bounding wrapper cells) are always shifting in an INTEST mode (and EXTEST mode) during test (both scan-shift and functional capture).

Note: This option can only be specified when inserting the 1500 wrapper cells using the `-shared_through` approach.

`-output_select_cfi {`*`pin`*`|`*`port`*`}`

Specifies the source pin name of the decoded control logic for the `select_cfi` signal for output bounding wrapper cells.

Use this option to connect an already decoded signal. Otherwise, control decoder logic may have to be inserted for each cell, and extra control wires will have to be hooked up to each wrapper cell.

`-output_shared_threshold` *`integer`*

Specifies the maximum number of scannable flops that can be shared in wrapper cells. When the number of scannable flops exceeds the specified threshold value, a single dedicated wrapper cell will be inserted for the output port.

Default: `10`

`-override_threshold_use_dedicated {`*`pin`*`|`*`port`*`}...`

Specifies the list of pins or ports for which a dedicated wrapper cell must be inserted.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-override_threshold_use_dedicated_string` *`string`*

Specifies the list of pins or ports as a string for which a dedicated wrapper cell must be inserted.

`-override_threshold_use_shared {`*`pin`*`|`*`port`*`}...`

Specifies the list of pins or ports for which a shared wrapper cell must be inserted.

`-override_threshold_use_shared_string` *`string`*

Specifies the list of pins or ports as a string for which a shared wrapper cell must be inserted.

`-preview` Shows a preview of the wrapper cell that would be inserted.

`-respect_dft_constants {test_setup | tied}`

Controls the propagation of constants and test signal values to prune the fanin (and fanout) paths of ports to find the sequential endpoints in the functional path for shared wrapper cell insertion. Possible values are:

`test_setup`: Propagates constant values and test_mode values

`tied`: Propagates constant (1'b0, 1'b1) values

If you omit this option, the command will perform full path tracing using a structural approach.

`-shared_through {buffer|inverters_and_buffers|combinational}`

Specifies whether the functional flop in the wrapper cell must be shared. If a shared cell is inserted, the command traces through the logic to identify a shareable functional flop (or flops).

A functional flop in a wrapper cell can be shared if

1. It is directly connected to the core pin through
   - a series of buffers (`buffer`)
   - a series of inverters and buffers
   - complex combination logic (`combinational`)
2. It is mapped to a scan flip-flop for DFT purposes.
3. The clock pin to the flop is controllable; that is, the flip-flop must pass the DFT rules.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

4. The flop has no connected enable pin.
5. The functional flop is not already shared with another wrapper cell.

Note: If you use this option and the functional flop cannot be shared, the RC-DFT engine issues a message and inserts a dedicated cell if you specified the `-wck` option, otherwise an error message is given.

`-skipped_locations_variable` *`Tcl_variable`*

Writes the locations where no wrapper cells can be inserted to the specified Tcl variable. If you omit this option, the command fails if you have any such locations specified.

`-use_power_domain_of_ports_for_dedicated`

Causes dedicated wrapper cells inserted at the top hierarchy to inherit the power domain of the port which is covered by the wrapper cell.

By default, the dedicated wrapper cell inherits the power domain of the hierarchy in which it is inserted.

`-wcap` *`driver`*

Specifies the capture control source pin or port name.

Note: This option is ignored if you specified the `-decoded_select_cfi` option.

`-wck` *`driver`*

Specifies the test clock source pin or port name. If specified, this pin will connect to the clock pin of the inserted dedicated wrapper cells.

Note: If this option is not specified, the tool will identify a local test clock to be used to connect to the clock pin of the inserted dedicated wrapper cells.

`-wext {`*`pin`*`|`*`port`*`|`*`constant`*`}`

Specifies the control source pin or port name, or constant value for outward facing test mode.

`-wig` *`driver`*

Specifies the in-guard control source pin or port name for an inward-facing wrapper cell.

Note: This signal will typically be controlled by the `-wext` control signal.

`-wint {`*`pin`*`|`*`hpin`*`|`*`port`*`|`*`constant`*`}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the control source pin or port name, or constant value for inward-facing test mode.

Note: This option is ignored if you specified the `-input_select_cfi` option.

`-wint_in_out {pin|hpin|port|constant}`

Specifies the control source pin or port name, or constant value for inward as well as outward facing test mode.

`-wog driver`

Specifies the out-guard control source pin or port name for an outward-facing wrapper cell.

Note: This signal will typically be controlled by the `-wint` control signal.

`-wrap_tied_constant_ports`

Specifies to insert dedicated wrapper cells on output ports driven by a logic 0 or logic 1 constant.

`-wsen_in driver`

Specifies the shift-enable signal for input bounding wrapper cells. Specify the shift-enable source pin or port name.

`-wsen_out driver`

Specifies the shift-enable signal for output bounding wrapper cells. Specify the shift-enable source pin or port name.

`-wsen_reuse_or_for_shared_wrapper`

Specifies to use a single top level or for shared wrapper cells.

Examples

- The following example inserts a dedicated wrapper cell for port `in[0]`.

```
insert_dft wrapper_cell -location in[0] -wsen_in WSEN -wsen_out WSEN \
-wint WINT -wck Wclk1
```

- The following example inserts four dedicated wrapper cells external to the hierarchical pins of the blackbox instance `hardmacro`. The two wrapper cells inserted for the hierarchical input pins are controlled in `INTEST` mode, while the two wrapper cells inserted for the hierarchical output pins are controlled in `EXTEST` mode. Because the `hardmacro` instance is modeled as a blackbox, you must specify the `-floating_location_ok` option to insert the wrapper cells.

```
insert_dft wrapper_cell -wsen_in WSEN -wsen_out WSEN-wck WCK -wint WINT \
    -wext WEXT -floating_location_ok \
    -location "hardmacro/pins_in/A hardmacro/pins_in/B \
      hardmacro/pins_out/X hardmacro/pins_out/Y"
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	connect_scan_chains on page 906
Sets this attribute:	core_wrapper

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft wrapper_instruction_register

```
insert_dft wrapper_instruction_register
     [-wrstn test_signal] [-wrck {pin|port}]
     [-selectwir test_signal]
     [-shiftwr test_signal]
     [-capturewr test_signal]
     [-updatewr test_signal]
     [-wsi port ] [-wso port]
     [-prefix string] [-design design]
```

Inserts a Wrapper Instruction Register (WIR) of length 3 and connects the Wrapper Control Signal, Wrapper Serial Input, and Wrapper Serial Output to the WIR.

In addition, the command generates the `SERIAL`, `PARALLEL`, `BYPASS`, `INTEST`, `EXTEST`, and `SHIFT_WBY` signals for the Wrapper Boundary Registers (WBR).

Options and Arguments

`-capturewr` *`test_signal`*
: Specifies the test signal used as capture signal for the Wrapper Instruction Register.

`-design` *`design`*
: Specifies the design in which you want to insert the wrapper instruction register. This option is required if multiple designs are loaded.

`-prefix` *`string`*
: Specifies the string to be prepended to the WIR name.

`-selectwir` *`test_signal`*
: Specifies the test signal used as select signal for the Wrapper Instruction Register.

`-shiftwr` *`test_signal`*
: Specifies the name of the shift signal for the Wrapper Instruction Register.

`-updatewr` *`test_signal`*
: Specifies the test signal used as update signal for the Wrapper Instruction Register.

`-wrck {`*`pin`*`|`*`port`*`}`
: Specifies the pin or port on which the Wrapper Instruction Register clock is defined.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-wrstn` *`test_signal`*

Specifies the test signal used as reset signal for the Wrapper Instruction Register.

`-wsi` *`port`*

Specifies the port used as Wrapper Serial Input.

`-wso` *`port`*

Specifies the port used as Wrapper Serial Output.

Related Information

Affected by this command: define_dft test_bus_port on page 985

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_dft wrapper_mode_decode_block

```
insert_dft wrapper_mode_decode_block [-name string]
     [-inside {instance|design}]
     [-create_wrapper_shift_enables_for_delay_test
          shift_enable]
     [create_cgic_enable_for_wrap shift_enable]
     [-directory directory] [-preview] [-design design]
```

Builds a 1500 mode decode block based on the scan modes that were defined with type `wrapper` for use with the IEEE 1500 core wrapper cells insertion. This module decodes the various modes needed for different scan configurations and also decodes the `EXTEST` and `INTEST` signals from these scan modes.

The IEEE 1500 wrapper insertion feature is documented in the *Genus Design for Test Guide for Legacy UI* - chapter Inserting Core Wrapper Logic.

Options and Arguments

`-create_cgic_enable_for_wrap` *`shift_enable`*

Specifies the shift-enable signal to be gated by the inverse of the decoded `EXTEST` signal. You can specify this gated signal (`DFTCGENI`) as the `lp_clock_gating_test_signal` to control the test pin of the functional clock gates which feed the core flops. After inserting the core wrapper logic, the test pins of the functional clock gates which feed the shared wrapper cells can be controlled by the `DFTWSEO` signal by rewiring the test pin connections.

`-create_wrapper_shift_enables_for_delay_test` *`shift_enable`*

Specifies the shift-enable signal to use to create two new shift enable signals: the wrapper shift-enable signal used for `INTEST` mode (gated by the decoded `INTEST` signal), and the wrapper shift-enable signal used for `EXTEST` mode (gated by the decoded `EXTEST` signal).

`-design` *`design`*

Specifies the design in which you want to insert the wrapper mode decode block. This option is required if multiple designs are loaded.

`-directory` *`directory`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Specifies the directory to which the generated decoder RTL file must be written. *Default:* `current_working_directory/1500`
`-inside` *`instance`*	
	Specifies the hierarchical instance in which the wrapper mode decode block must be instantiated. By default, the wrapper mode decode block is inserted in the top-level design.
`-name` *`name`*	Specifies the name of wrapper mode decode block. *Default:* `wrapperModeDecodeBlock`
`-preview`	Prints the RTL of the generated decode block to the screen.

Example

The following example inserts a wrapper mode decode block based on the defined wrapper dft_configuration_modes.

```
define_dft dft_configuration_mode -name functional -type wrapper -usage mission \
    -mode_enable_low TS1 TS2 TS3 TS4
define_dft dft_configuration_mode -name serialIntest -type wrapper -usage intest \
    -mode_enable_low TS4 TS3 TS2 -mode_enable_high TS1
define_dft dft_configuration_mode -name serialExtest -type wrapper -usage extest \
    -mode_enable_low TS4 TS3 TS1 -mode_enable_high TS2
define_dft dft_configuration_mode -name parallelIntest -type wrapper
    -usage intest -mode_enable_low TS4 TS3 -mode_enable_high TS2 TS1
define_dft dft_configuration_mode -name parallelExtest -type wrapper \
    -usage extest -mode_enable_low TS4 TS2 TS1 -mode_enable_high TS3
define_dft dft_configuration_mode -name parallelCompress -type wrapper \
    -usage intest -mode_enable_low TS4 TS2 -mode_enable_high TS3 TS1

insert_dft wrapper_mode_decode_block \
    -create_wrapper_shift_enables_for_delay_test DFTWSE
```

Related Information

Affects these commands:	concat_scan_chains on page 897
	connect_scan_chains on page 906
Affected by this command:	define_dft dft_configuration_mode on page 925

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_test_compression

```
insert_test_compression
     [ -use_existing_channels actual_scan_chain...]
     | -build_new_scan_chains integer
       [-asymmetric_scan_ins integer]]
     [-decompressor {broadcast | xor }]
     [-mask_sharing_ratio integer]
     [ -compressor xor [-mask {wide1|wide2}]
     | -compressor hybrid [-mask {wide0|wide1|wide2}]]
     [ -use_existing_wrapper_channels actual_scan_chain...
       [-use_wir_macro instance] [-bypass_reg]]
     [-auto_create] [-inside instance] [-top]
     [-dont_map] [design]
```

Inserts the test logic infrastructure for scan compression and hierarchical test. This includes inserting and configuring the compressor, de-compressor and x-masking logic, and connecting them to existing or newly created compression mode scan chains (also called STUMPS channels). The DFT logic can be optionally controlled from instructions decoded from the core's IEEE 1500 wrapper instruction register (WIR).

Options and Arguments

`-asymmetric_scan_ins` *integer*

Specifies the number of scan data input pins for the compression macro. You only need to specify this option if the number differs from the number of scan data output pins.

Default: The number of scan chains specified with the `-build_new_scan_chains` option.

`-auto_create`

Automatically creates the necessary compression control signals as top-level ports.

Note: You can also define the test bus ports using the `define_dft test_bus_port` command.

`-build_new_scan_chains` *integer*

Specifies the total number of top-level scan chains that is desired.

`-bypass_reg`

Inserts a single-bit bypass register to the scan paths in 1500 bypass mode for the hierarchical test flows.

Note: This option can only be specified with the `-use_existing_wrapper_channels` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-compressor {xor | hybrid}`

Specifies the type of compression logic to be built:

- `xor` specifies to build an XOR-based compressor
- `hybrid` specifies to build a `MISR` compression with `MISR` bypass capability. Bypassing the MISR allows you to perform compression using just the XOR compressor.

Default: `xor`

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic to be built:

- `xor` specifies to build an XOR-based spreader network in addition to the broadcast-based decompression logic
- `broadcast` specifies to build a broadcast-based decompression logic (simple scan fanout).

Default: `broadcast`

`design`

Specifies the name of the top-level design.

`-dont_map`

Prevents the inserted macros from being mapped even if the design is already mapped to the target library.

`-inside` *`instance`*

Specifies the hierarchical instance in which to instantiate the compression logic.

`-mask {wide0|wide1|wide2}`

Inserts scan channel masking logic of the specified type.

The masking types that can be used depend on the compressor type specified with the `-compressor` option.

By default, no masking logic is inserted.

`-mask_sharing_ratio` *`integer`*

Specifies the number of internal scan channels sharing a mask register.

`-top`

Specifies to insert the hierarchical compression logic at the top level.

It is used for top-level compression and TIM insertion in a hierarchical test flow after the lower-level blocks have been synthesized, DFTs are inserted and scan abstract created. It connects the blocks together via the top TIM.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-use_existing_channels` *`actual_scan_chain...`*

Specifies the names of the actual scan chains to be used as compression channels.

`-use_existing_wrapper_channels` *`actual_scan_chain...`*`]`

Specifies the names of the actual scan chains to be used as wrapper compression channels.

`-use_wir_macro` *`instance`*

Specifies the path name of the wrapper instruction register (WIR) instance.

Note: This option can only be specified with the `-use_existing_wrapper_channels` option.

Related Information

Related command: define_dft test_bus_port on page 985

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

map_dft_unmapped_logic

```
map_dft_unmapped_logic
     [-update_placement][-map_to_scan]
     {design|module}
```

Maps any logic that was added by previous DFT commands, but not mapped at the time of the insertion. For example, logic added by test point insertion, scan chain connection, and so on.

Options and Arguments

{design|module}	Specifies the name of the design or the module for which the added DFT logic should be mapped.
`-map_to_scan`	Specifies to map flops to their scan counterparts.
`-update_placement`	Specifies to also update the placement of the added DFT logic.

Related Information

Related commands:

add_opcg_hold_mux on page 851
insert_dft boundary_scan on page 1026
insert_dft jtag_macro on page 1046
insert_dft pmbist on page 1065
insert_dft ptam on page 1071
insert_dft shadow_logic on page 1077
insert_test_compression on page 1106
insert_dft test_point on page 1083
replace_opcg_scan on page 1121
fix_dft_violations on page 1010

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

map_pmbist_cgc_to_cgic

```
map_pmbist_cgc_to_cgic
     [-design design] [-module_prefix_list string]
     [-clock_gating_cell lib_cell]
```

Maps programmable MBIST clock gating logic to integrated clock gating cells.

Options and Arguments

`-clock_gating_cell` *`lib_cell`*

Specifies the name of the `latch_posedge_precontrol` clock gating cell to be used. This option has the highest priority.

The libcell specified with this option should have the attribute `clock_gating_integrated_cell` set equal to `latch_-posedge_precontrol`.

If this option is not specified with the command, then the libcell is retrieved from the attribute `lp_clock_gating_cell` set on the module containing the clock gating instance. If such a libcell is found, then that libcell is used, otherwise libraries are searched to find a suitable `lp_clock_gating_cell` which has the attribute `clock_gating_integrated_cell` set equal to `latch_posedge_precontrol`.

`-design` *`design`*

Specifies the name of the top-level design.

`-module_prefix_list` *`string`*

Specifies a list of prefix strings provided to `add_pmbist` command at various levels.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

map_pmbist_ffn

```
map_pmbist_ffn
     [-design design]
```

Maps programmable MBIST negative active flop modules to proper library cells.

Options and Arguments

`-design` *`design`*	Specifies the name of the top-level design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

map_pmbist_ffsync

```
map_pmbist_ffsync
     [-design design]
```

Maps programmable MBIST synchronizer flop modules to proper library cells.

Options and Arguments

`-design` *`design`*	Specifies the name of the top-level design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_dft_abstract_model

```
read_dft_abstract_model
     [-ctl [-use_scan_structures_se_only]
          [-override_scan_libcell] [-advanced]]
     [-segment_prefix string]
     [-instance instance]
     [-assume_connected_shift_enable] file
```

Reads in the scan abstract model of a design that is used as a core or IP block in the current design. The scan abstract model defines the scan chain architecture of the subdesign and is used as scan chain segments in the configuration of the top-level scan chains of the current design.

The extracted scan chain information is stored in:

`/designs/top_design/dft/scan_segments`

Options and Arguments

`-advanced`

Instructs the CTL reader to parse the IsConnected constructs for feedthrough pins and apply these to the design as dft_controllable pins. In addition, any signal listed as a DataType Constant will be defined as a test mode signal with its ExpectState value used as the active test mode.

Note: This option must be specified with the `-ctl` option

Note: The writing of feedthrough pins to the scan abstract models using `dft_controllable` construct in native abstract model, and `IsConnected` construct in CTL abstract model is controlled using the `dft_include_controllable_pins_in_abstract_model` root attribute. The writing of signals as constant values in the native abstract and CTL abstract model is controlled through the `dft_include_test_signal_outputs_in_abstract_model` root attribute.

`-assume_connected_shift_enable`

Indicates that the shift-enable port specified in the DFT abstract model for the block being read in is already connected to logic external to this block. Therefore the scan configuration engine does not need to modify the existing connection.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: If you specify this option and the shift-enable pin is *not* connected, the scan configuration engine will *not* make the connection.

If you do not specify this option, the scan configuration engine will make the connection to the shift-enable port specified in the DFT abstract model. If a connection already existed, it will be first removed.

`-ctl`

Specifies that the scan abstract model was written using the Core Test Language (CTL) format (IEEE format `P1450.6`).

If you omit this option, the scan abstract model is assumed to consist of a list of `define_dft_abstract_segment` commands, one scan segment per scan chain in the subdesign.

`file`

Specifies the file that contains the abstract model description.

`-instance` *`instance`*

Applies the scan abstract model to the specified hierarchical instance.

If you read in an abstract model written in native RC format, this instance must be an instantiation of the subdesign specified through the `-module` option in the abstract model.

If you read in an abstract model written in CTL format, this instance must be an instantiation of the subdesign specified in the `Environment` section of the CTL file.

If this option is omitted, the scan abstract model is applied to all instances of the subdesign.

`-override_scan_libcell`

Allows you to redefine the `test_cell` definition of a scan cell in the Liberty library using the test information applied from the CTL file.

For example, you can use this option to update the sequential length of the scan primitive from a single bit to multi-bit cell when the intended usage of this cell is a multi-bit scan flop synchronization cell.

Note: This option must be specified with the `-ctl` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-segment_prefix` *`string`*

Adds the specified string as a prefix to the

- Segment name defined in the native RC format file
- Chain name defined in the CTL file

`-use_scan_structures_se_only`

Indicates that the shift-enable signal for an abstract segment must be read from its `ScanChain` definition in the `ScanStructures` block in the CTL file.

If a shift-enable signal is not specified, the abstract segment is created using the `-connected_shift_enable` option of the `define_dft abstract_segment` command.

Note: This option must be specified with the `-ctl` option.

Related Information

Affected by these commands:	check_dft_rules on page 870
	connect_scan_chains on page 906
Related commands:	write_dft_abstract_model on page 1162
	write_hdl on page 404 (`-abstract`)
Sets these attributes:	Scan Segment Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_io_speclist

`read_io_speclist` *`iospeclist_file`*

Reads in the specified IOSpecList input file to be used for boundary scan insertion.

The IOSpecList input file is only required to provide information that cannot be inferred from the design, and the command-line options of the `insert_dft boundary_scan` command.

You need an IOSpecList input file if

- The I/O pad cells in your library do not use the standard pin names
- Your design has pin sharing logic to shared functional output signals that was inserted before you insert boundary scan logic
- You want to customize the location of the boundary cells in the boundary register

You can also use an IOSpecList input file if

- You want to use custom boundary cells
- You want to use user-defined TAP instructions (such as those required for MBIST or PTAM) and use specific opcodes specified using `JTAG_Inline` syntax

  Note: You can also use the `define_dft jtag_instruction` command to enter user-defined instructions.

Options and Arguments

`iospeclist_file`	Specifies the IOSpecList input file.

Related Information

Refer IOSpecList File Format Specifics for Boundary Scan in chapter Inserting Boundary-Scan Logic of the *Genus Design for Test Guide for Legacy UI*

Affects these commands: insert_dft boundary_scan on page 1026
write_io_speclist on page 1199

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_memory_view

```
read_memory_view
     {-cdns_memory_view_file file
     | -preview -ctl file [-directory string]
     | -preview [-directory string]
       [-mbif file [-mbif_setup_file file]
       [-mbif_modus_tracing_setup_file file]]}
     [-save_all_memory_views] [-update_existing_views] [design]
```

Loads the memory and macro views file for the memories contained within the design.

With -`preview` option, it generates the memory and macro view file template to be corrected and read back in for the memories and macros contained within the design. It also converts the memory views specified in Core Test Language (CTL) to CDNS specific format.

Without -`preview` option, it loads the memory an macro views file for the memories contained within the design.

The `cdns_memory_view_file` can supply memory views containing only the module sections for a design which include all targeted memory port descriptions, address range, read access delay, physical information including data bit order, write mask assignment and memory cell array layout, redundant features and wrapper information. The `cdns_memory_view_file` may also supply macro views containing macro interface definitions. Macro views provide a means to specify shared test bus interface to be used for memory testing, logical memories that can be accessed by a particular interface, physical memories contained within logical memories and their access mechanism. After the view file is loaded, Genus updates the design hierarchy with the views information and prints summary tables.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-cdns_memory_view_file` *file*

Specifies the file containing the physical view of the memories and macros in Cadence format. This file can overload the .lib file, providing information that is missing in the .lib files.

`-ctl` *file*

Specifies a file written using the Core Test Language (CTL) format (IEEE Std 1450.6.2-2014) that contains information on memory modeling.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

design Specifies the name of the top-level design.

-directory *string* Specifies the directory where the template file must be generated.

Default: ./pmbist_*design*

-mbif *file* Specifies a customer-specific MBIST information file for use in generating macro views when the -preview option is specified.

This option must be specified with the -mbif option.

-mbif_modus_tracing_setup_file *file*

Specifies the name of a Tcl file that contains the Modus commands to perform circuit tracing and determine the logical to physical mapping of the memories. Modus has many different tracing options.

The default tracing option values are chosen to complete tracing successfully. You can override these values using the commands in the specified file.

-mbif_setup_file *file*

Specifies the name of a Tcl file that contains the Genus commands to be used during the generation of the macro view.

This option must be specified with the -mbif option.

-preview Requests the generation of a template file for the memory or macro views.

-save_all_memory_views

Saves all memory views even if they are not used in the design.

Specify this option if the design contains macros which are black boxes and physical memories reside inside these macros. Normally memory views are only retained within Genus if they are detected within the design. As these memory views must be read prior to reading macro views and memories are not visible within black-boxed macros, the use of this option ensures that all the required memory views are available to support macro view physical memory bindings

-update_existing_views

Specifies to update existing memory views

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	insert_dft pmbist on page 1065
Related attributes:	Memory Data Bit Structure Attributes
	Memory Libcell Attributes
	Memory Libpin Action Attributes
	Memory Libpin Alias Attributes
	Memory Spare Column Attributes
	Memory Spare Column Map Address Attributes
	Memory Spare Column Map Data Attributes
	Memory Spare Row Attributes
	Memory Spare Row Map Address Attributes
	Write Mask Bit Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_pmbist_interface_files

```
read_pmbist_interface_files
     -directory string {design|subdesign|base_cell|libcell}
```

Reads interface files for Programmable Memory Built-In-Self-Test (PMBIST) generated by an earlier call to the `write_pmbist_interface_files` command correlated to a particular *design* or *subdesign*.

These interface files represent an abstract model of the previous PMBIST insertion process, supporting not only a bottom-up flow for incremental PMBIST insertion but also the generation of patterns to exercise the PMBIST logic from Modus `create_embedded_test` command.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

{*design*|*subdesign*|*base_cell*|*libcell*}	Specifies the name of the design, subdesign, base_cell, or libcell for which the interface files are read.
-directory *string*	Specifies the directory that contains the interface files.

Related Information

Refer to PMBIST Interface Files in the *Genus PMBIST Guide for Legacy UI*.

Affects this command:	insert_dft pmbist on page 1065
Related commands:	write_pmbist_interface_files on page 1207

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

replace_opcg_scan

```
replace_opcg_scan
     -edge_mode test_signal
     [-dont_map] [-effort {high | low}] [design]
```

Replaces domain crossing scan flops with their `OPCG`-equivalent flops by adding a hold mux and inverter (referred to as a toggle mux) in the scan data path. This command is used for designs with OPCG logic when the OPCG domain blocking attribute is enabled in the flow. If the library includes OPCG scan flops with the toggle mux logic built into the liberty cell, you should specify these scan flops using the `set_opcg_equivalent` command ahead of the `replace opcg_scan` command to use these cells as the OPCG-equivalent flops. The `replace opcg_scan` command should be run after the test compression logic has been added to the design.

Options and Arguments

design
: Specifies the design in which you want to replace domain-blocking scan flip-flops.

`-dont_map`
: Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

`-edge_mode` *test_signal*
: Specifies the global edge-mode signal to connect.

`-effort {high | low}`
: Specifies the approach to inserting and mapping the toggle mux and inverter logic to be inserted for the `OPCG` domain crossing flops.

- Using a `low` effort level, the toggle mux and inverter logic will be mapped directly to gates from the target technology library and connected in the design.

  Use this approach when you have thousands of at-speed domain crossing flops to be processed as it offers better runtime at the expense of timing accuracy.

- Using a `high` effort level, the toggle mux and inverter logic will be mapped to generic components first, and the newly added logic will then be mapped to gates from the target technology library.

Default: `high`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by these commands: reset_opcg_equivalent on page 1138
set_opcg_equivalent on page 1142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

replace_scan

```
replace_scan [-to_non_scan]
     [-dont_check_dft_rules] [design]
```

This command either

- Replaces non-scan flops with their scan-equivalent flip-flops if the design was previously mapped.

  In this case, the `dft_scan_map_mode` design attribute must be set to either `tdrc_pass` or `force_all`. If set to `tdrc_pass`, you must have run the DFT rule checker.

- Replaces all scan flops that are part of shift register segments with non scan flops except for the first element of each shift register segment.

This command should be run on the mapped design and prior to building the scan chains in either a logical or physical flow. The presumption being that the scan elements will be properly mapped for DFT and included into the scan chains before physical synthesis is performed on the design.

Options and Arguments

design	Specifies the design in which you want to replace regular flip-flops.
-dont_check_dft_rules	Prevents the DFT rules from being automatically checked.
-to_non_scan	Replaces all scan flops that are part of shift register segments to non scan flops except for the first element in the segments.

Related Information

Affected by these commands:

check_dft_rules on page 870
identify_test_mode_registers on page 1021
identify_test_mode_registers on page 1021
set_scan_equivalent on page 1144

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Affected by this attribute: dft_scan_map_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_chains

Refer to report dft_chains in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_clock_domain_info

Refer to report dft_clock_domain_info in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_core_wrapper

Refer to report dft_core_wrapper in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_registers

Refer to report dft_registers in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_setup

Refer to report dft_setup in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report dft_violations

Refer to report dft_violations in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report opcg_equivalents

Refer to report opcg_equivalents in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report scan_compressibility

Refer to report scan_compressibility in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report test_power

Refer to report test_power in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_longest_wires

Refer to report_chains_longest_wires in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_total_wirelength

Refer to report_chains_total_wirelength in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_chains_wires

Refer to report_chains_wires in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_scan_compression_logic

Refer to report_scan_compression_logic in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_opcg_equivalent

`reset_opcg_equivalent [libcell]...`

Removes the specified scan library cells from the `OPCG`-equivalence table which was previously defined using a (number of) `set_opcg_equivalent` command(s).

If you do not specify any library cells, the command removes all `OPCG`-equivalent mappings.

Options and Arguments

`libcell`	Specifies a scan library cell to be removed from the `OPCG`-equivalence table.

Example

The following example removes the `snl_ffqx1` cell from the `OPCG`-equivalence table.

```
reset_opcg_equivalent snl_ffqx1
```

Related Information

Affects this command:	replace_opcg_scan on page 1121
Related command:	set_opcg_equivalent on page 1142

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_scan_equivalent

`reset_scan_equivalent [libcell]...`

Removes the specified non-scan library cells from the scan-equivalence table which was previously defined using a (number of) `set_scan_equivalent` command(s).

If you do not specify any library cells, the command removes all scan-equivalent mappings.

Options and Arguments

libcell	Specifies a non-scan library cell to be removed from the scan-equivalence table.

Example

The following example removes the `snl_ffqx1` cell from the scan-equivalence table.

```
reset_scan_equivalent snl_ffqx1
```

Related Information

Affects this command:	replace_scan on page 1123
Related command:	set_scan_equivalent on page 1144

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_compatible_test_clocks

```
set_compatible_test_clocks
     {-all | -none | list_of_test_clocks}
     [-dont_check_dft_rules] [-design design]
```

Specifies the compatible test clocks whose related scan flip-flops can be merged into a single scan chain using lockup elements in between. By default, no test clocks are assumed compatible unless they are defined as independent test clocks in the same test clock domain.

Note: This command applies only to the muxed scan style.

Test clocks that are declared compatible belong to the same DFT clock domain.

Important

Only those test clocks with the same clock period can be made compatible as independent test clocks in the same test clock domain.

Options and Arguments

`-all`	Specifies that all test clocks are compatible.
`-design` *`design`*	Specifies the design for which you want to specify compatible test clocks.
`-dont_check_dft_rules`	Prevents the DFT rules from being checked automatically after specifying the compatible test clocks.
`list_of_test_clocks`	Specifies the compatible test clocks. You must specify the test clock object name. **Note:** To allow combining flip-flops from the same DFT domain—which are triggered by either edge of the same test clock—on the same scan chain, you need to set the `dft_mix_clock_edges_in_scan_chains` root attribute to `true`. By default, only same edge clocks are mixed.
`-none`	Specifies that none of the test clocks are compatible.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	connect_scan_chains on page 906
Related attribute:	dft_mix_clock_edges_in_scan_chains

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_opcg_equivalent

```
set_opcg_equivalent
     -scan_cell libcell -opcg_cell libcell
     [-tieoff_pins string] [-pin_map list_of_pin_groups]
     -edge_mode_pin string -loop_back_pin string
```

Controls the `OPCG`-equivalent cell type that is used during the conversion of a scan flip-flop to an `OPCG` cell by the `replace_opcg_scan` command.

The command creates an `OPCG`-equivalence table.

An `OPCG` cell is a special scannable cell with an additional (hold) mux in its scan-data path. Depending upon the value of the `-edge_mode_pin`, the mux will either circulate or loopback the inverted value of its output pin to the `-loop_back_pin`, or capture the scan-in data from the scan-data path.

Options and Arguments

`-edge_mode_pin` *`pin`*	Specifies the edge mode pin of the `OPCG` cell to connect to.
`-loop_back_pin` *`pin`*	Specifies the loopback pin of the `OPCG` cell to connect to.
`-opcg_cell` *`libcell`*	Specifies the `OPCG` library cell to map to.
`-pin_map` *`list_of_pin_groups`*	Indicates how to map a pin from the scan flop to a pin in the `OPCG` cell when the pin names in the cells do not match. The *`list_of_pin_groups`* has the following format: `{{`*`scan_pin opcg_pin`*`} {`*`scan_pin opcg_pin`*`}...}`
`-scan_cell` *`libcell`*	Specifies a scan flip-flop library cell to be replaced.
`-tieoff_pins` *`string`*	Specifies the tie-off value for extra pins on the `OPCG` cell. The *`string`* has the following format: `{{`*`pin tie_off_value`*`} {`*`pin tie_off_value`*`} ...}` The value can be a logic `0` or `1`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following example assumes that the pin names in the scan and `OPCG` flip-flops match, and that there are no extra pins in the `OPCG` flop to be tied off.

```
set_opcg_equivalent -scan_cell SDFFQ_X1M -opcg_cell S2DFFQQN_X1M \
    -edge_mode_pin TEL -loop_back_pin TI
```

Related Information

Affects this command:	replace_opcg_scan on page 1121
Related command:	reset_opcg_equivalent on page 1138

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_scan_equivalent

```
set_scan_equivalent
     -non_scan_cell libcell -scan_cell libcell
     [-tieoff_pins string] [-pin_map list_of_pin_groups]
```

Controls the scan-equivalent cell type that is used during the conversion of a non-scan flip-flop which passes the DFT rule checks to a scan flop. Use the `replace_scan` command to perform the actual conversion to scan.

Note: The Genus-DFT engine automatically derives the scan data input, scan data output, and other test signals from the `test_cell` description of the scan flop in the target library.

In most cases Genus will automatically identify scan-equivalent cells in the library from the function definition in the `test_cell` section. However, if the library description is incomplete this command can be used to force equivalence.

Options and Arguments

`-non_scan_cell` *`libcell`*

Specifies a non-scan flip-flop library cell.

`-pin_map` *`list_of_pin_groups`*

Indicates how to map a pin from the non-scan flop to a pin in the scan flop when the pin names in the cells do not match.

The *`list_of_pin_groups`* has the following format:

`{{`*`non_scan_pin scan_pin`*`} {`*`non_scan_pin scan_pin`*`}...}`

`-scan_cell` *`libcell`*

Specifies a scan flip-flop library cell.

`-tieoff_pins` *`string`*

Specifies the tie-off value for extra scan cell pins.

The *`string`* has the following format:

`{{`*`pin value`*`} {`*`pin value`*`} ...}`

The value can be a logic `0` or `1`.

Example

- The following example assumes that the pin names in the non-scan and scan flip-flops match, and that there are no extra pins in the scan flop to be tied off.

```
set_scan_equivalent -non_scan_cell snl_ffqx1 -scan_cell snl_sffqx1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	replace_scan on page 1123
Related command:	reset_scan_equivalent on page 1139

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

update_scan_chains

```
update_scan_chains
     [-flops instances] [-chains actual_scan_chains]
     [-type {dfa|rrfa|user|compression_testpoints}]
     [-max_list_flops integer]
     [-preview] [design] [> file]
```

Inserts the identified scannable test points in the existing scan chains. The scan chains must be connected before inserting these test points.

Note: Test points can only be added to scan chains that have not been compressed.

Options and Arguments

`-chains` *`actual_scan_chains`*

Specifies the names of the actual scan chains to be updated

By default, all actual scan chains are updated.

`design`

Specifies the design in which you want to update the scan chains.

`-flops` *`instances`*

Specifies the test point instances to be added.

This option can't be used together with the `-type` option.

`-max_list_flops` *`integer`*

Specifies the maximum number of test point flops to report.

By default, all test points will be reported.

`-preview`

Reports what would be done, but does not update the scan chains

`-type {dfa|rrfa|user|compression_testpoints}`

Specifies the type of test points to add to the scan chains.

Test points can be inserted after DFA or RRFA analysis or can be user-inserted.

This option can't be used together with the `-flop` option.

By default, all test point types are inserted.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command shows a preview of the scan chains after they would be updated with the test points inserted by the DFA analysis.

```
update_scan_chains -type dfa -preview \
    -chains [find /des*/DLX_CORE -actual_scan_chains *]
```

Related Information

Affected by these commands:	add_analyzed_test_points on page 848
	add_opcg_hold_mux on page 851
	insert_dft test_point on page 1083
	insert_dft user_test_point on page 1089
Related command:	connect_scan_chains on page 906

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_atpg

```
write_atpg
     { -cadence [-compression | > file]
     | -mentor
     | -stil [-dft_configuration_mode dft_config_mode]
       [> file]}
     [-decimals_ok] [-picoseconds]
     [-test_clock_waveform test_clock]
     [-apply_inputs_at integer]
     [-apply_bidirs_at integer]
     [-strobe_outputs_at integer]
     [-strobe_width integer] [design]
```

Writes out the scan-chain information for an Automatic Test Pattern Generator (ATPG) tool in a format readable by the designated ATPG tool.

The ATPG tool uses this information to generate appropriate test patterns. The file extension given to the interface file(s) is determined by the selected tool.

The interface file is useful only to the third-party tool if the test synthesis tool has connected the scan chain. Therefore, you should use this command only if the test synthesis tool has connected the scan chains.

Options and Arguments

`-apply_bidirs_at` *integer*

Specifies when in the test clock cycle to apply the bidirectional signals. Specify this time as a percentage of the test clock period.

Default: Same value as specified for `-apply_inputs_at`

`-apply_inputs_at` *integer*

Specifies when in the test clock cycle to apply the input signals. Specify this time as a percentage of the test clock period.

Default: `0`

`-cadence`

Creates pin-assignment files that capture the top-level scan-related signals (shift-enable, test-mode, test-clock and scan data IOs) for use by the Cadence ATPG tool.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you use this option without the `-compression` option, the command writes out the pin-assignment information for full scan mode only. The information is written to the specified file.

`-compression`

Creates pin-assignment files for full scan mode, compression mode and XOR decompression mode. The following files are generated: *`topmodulename`*`.FULLSCAN.pinassign`, *`topmodulename`*`.COMPRESSION.pinassign`, and *`topmodulename`*`.COMPRESSION_DECOMP.pinassign`.

Note: This option is only valid with the `-cadence` option.

`-decimals_ok`

Writes out decimal numbers. By default, time values are rounded off to integer numbers because many ATPG tools do not accept decimal numbers for test waveform time values. Use the `-picoseconds` option to minimize round-off errors.

`design`

Specifies the top module for which to write ATPG.

`-dft_configuration_mode` *`dft_configuration_mode`*

Writes ATPG information for the specified scan mode name.

Note: This option is only valid with the `-stil` option.

`file`

Specifies the file to which the output must be written.

If no file is specified, the output is written to standard out (`stdout`) and to the log file.

Note: This argument is only valid with the `-stil` and `-cadence` options.

`-mentor`

Creates an interface file in the format used by the Mentor Graphics ATPG tool. Files generated:

- `top_module.testproc`
- `top_module.dofile`

`-picoseconds`

Specifies to use picoseconds for the time unit. Use this option to minimize the round-off errors when rounding-off test waveform time values to integers.

Default: `nanoseconds`

`-stil`

Creates an interface file in the IEEE Standard Test Interface Language (STIL) format (IEEE format 1450.1).

Note: The generated STIL format is TetraMAX compatible.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-strobe_outputs_at` *`integer`*

Specifies when in the test clock cycle to strobe the outputs. Specify this time as a percentage of the test clock period.

Default: 40

`-strobe_width` *`integer`*

Specifies how long the outputs are valid during the test clock cycle. Specify this time as a percentage of the test clock period.

Default: 0

`-test_clock_waveform` *`test_clock`*

Specifies to use the clock waveform of the specified test clock.

Default: first test clock object found

Related Information

Affected by this command:	compress_scan_chains on page 882
	connect_scan_chains on page 906
	define_dft scan_clock_a on page 972
	define_dft scan_clock_b on page 975
	define_dft test_clock on page 990
Affected by these attributes:	Actual Scan Chain Attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_bsdl

```
write_bsdl
     [-pinmap_file file]
     [-bsdl_package_name files]
     [-bsdlout file]
     [-include_private_instructions]
     [-expose_ports_with_pinmap]
     -directory string [-design design]
```

Generates a file describing the boundary scan architecture of the design in Boundary Scan Description Language (BSDL), along with two VHDL package files, `STD_1149_1_2001` and `CDNDFT_1149_1_2001`, which contain the supported boundary cell descriptions that were used during boundary scan insertion.

Note: Dedicated test-related signals (such as shift-enable, and test-mode signals defined without the `-shared_in` option) are also written to the BSDL file along with their respective compliance enable values.

The BSDL file is required by board and system test software to test the interconnects between chips and modules. It is also used by several applications in the Modus DFT Software product such as Boundary Scan Verification (BSV) and MBIST insertion and verification. More details are available in the section titled Writing a BSDL File in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-bsdl_package_name files`

Specifies the name of a VHDL file or a comma-separated list of VHDL files, each containing one or more custom boundary cell descriptions that were used during boundary scan insertion.

The name of each package file is added to the BSDL file in a `use` statement.

Note: This option is required if you used custom boundary cells in the design.

`-bsdlout file`

Specifies the name of the BSDL file to be generated.

If you omit this option, the output file is named using the `topmodulename.bsdl`.

`-design design`

Specifies the design for which to generate the BSDL file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-directory` *`string`* Specifies the directory to which the output files must be written.

`-expose_ports_with_pinmap`

Specifies to only expose functional and test ports with package pinmap information to the output BSDL file.

Additionally, this option prevents the writing of other ports connected in the boundary-scan register without package pinmap information. The names of these other ports will not appear in the `BOUNDARY_REGISTER` section of the BSDL file. Rather, these port names will be represented with an asterisk (`*`) and their associated boundary-scan cells will be represented as `INTERNAL` as shown in the following BSDL snippet:

```
attribute BOUNDARY_REGISTER of test: entity is
            ...
"8  (BC_OUT, *, INTERNAL, X)," &
```

When the `-expose_ports_with_pinmap` is not specified, all ports in the design will be written to each relevant section of the BSDL file, regardless of whether any pinmap information has been provided for any ports.

Package pinmap information may be provided during boundary-scan insertion or when writing the BSDL file using the `-pinmap_file` option.

`-include_private_instructions`

Specifies to include register access information for private instructions in the BSDL file.

`-pinmap_file` *`file`* Specifies the name of the pinmap file to be used to create a BSDL file.

Note: This file can have fewer pin-to-pad bonding requirements than the pinmap file specified for the boundary scan insertion.

Example

- The following command creates a BSDL file named `bsdlout`. The name of the package file for the custom boundary cells is added to the BSDL file.

  ```
  write_bsdl -directory .-bsdl_package_name MY_BIDIR_PKG -bsdlout bsdlout
  ```

  This causes the following line to be added to file `bsdlout`:

  ```
  use MY_BIDIR_PKG.all ;
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by this command:	insert_dft boundary_scan on page 1026
Related constraints:	define_dft shift_enable on page 978
	define_dft test_clock on page 990
	define_dft test_mode on page 995

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_compression_macro

```
write_compression_macro
     {-chains integer | -scan_in integer -scan_out integer}
     -sub_chains integer
     [-decompressor {broadcast | xor }]
     [-compressor
      { xor [-scan_in_pipeline_depth integer]
            [-scan_out_pipeline_depth integer]
            [-compressor_pipeline_depth integer]
      | opmisr | hybrid
      | smartscan_xor -smartscan_ratio integer
        [-smartscan_no_update_stage] [-smartscan_serial_only]
        [-smartscan_pulse_width_multiplier {1|2|4}] }]
     [-mask {wide0 | wide1 | wide2}]
     [-mask_sharing_ratio integer ]
     [-no_fullscan_muxing] [-jtag_control]
     [-serial_misr_read]
     [-shared_scan_in_pins string
       -asymmetrical_scan_in integer]
     [-block_select] [-compressed_chains integer]
     [-separate_mask_ports]
     [-remove_cancelling_terms_for_hierarchical_flow]
     [-low_pin_compression] [-bitwise_compressor]
     [-file file] [-info_file file]
```

Generates the RTL for a customized scan compression macro.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-asymmetric_scan_in` *integer*

Specifies the number of scan in pins for an asymmetric compression macro.

This option is only supported with `wide1` and `wide2` types of channel masking.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

For wide1 type of masking, the number of scan data input pins must equal the number specified for the `-chains` option minus one (*number_of_chains*-1), while for wide2 type of masking, the number must equal the specified number of chains minus two (*number_of_chains*-2).

`-bitwise_compressor`

Generates a compressor with bitwise xoring of the channels (that is, each channel feeds only a single SO).

Use this option for leaf or mid-level blocks during hierarchical compression to avoid simulation issues.

`-block_select`

Adds additional logic to bypass the block for compression inserted at the block level. In compression mode, the scan outputs are forced to zero. In uncompressed scan mode, the scan outputs are fed directly from the scan inputs.

An extra `BLOCK_SELECT` pin—to control the additional block select logic—is added to the compression macro.

If you specify this option with the `-separate_mask_ports` option, the mask registers are also bypassed by feeding the `MASK_OUT` ports directly from the `MASK_IN` ports.

Note: The `BLOCK_SELECT` pin should be held low to bypass the block and should be held high otherwise.

`-chains` *integer*

Specifies the number of top level scan data input/scan data output pairs. Typically, this is the number of uncompressed scan chains. For MISR-based compression, the `-chains` option must be greater than or equal to 16.

You cannot specify this option when you specify the `-scan_in` and `-scan_out` options. When these options are specified, the number of chains equals the lesser of the following: the scan_in plus the number of shared control pins, or scan_out.

`-compressed_chains` *integer*

Specifies the number of scan chains in lower-level blocks that are already compressed.

The command adds the `CCHAN_SI` and `CCHAN_SO` ports to the macro to connect to the compressed chains at a lower block directly. The `CCHAN_SO` port feeds the channel data directly into the compressor.

This option cannot be specified when the `-compressor` option is set to `smartscan_xor`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-compressor {xor | opmisr| hybrid | smartscan_xor}`

Specifies the type of compression logic to be built:

- `xor` specifies to build an XOR-based compressor
- `opmisr` specifies to build a `MISR`-based compressor
- `hybrid` specifies to build `MISR` compression with `MISR` bypass capability to effectively result in an XOR-based compressor.
- `smartscan_xor` specifies to add smartscan logic to the XOR-based compression macro.

Default: `xor`

`-compressor_pipeline_depth` *`integer`*

Specifies the number of pipeline stages to be inserted within the compressor block between the last flop in the scan channels and the compression macro output ports.

Note: Applies only to an XOR-based compressor.

`-decompressor {broadcast | xor}`

Specifies the type of decompression logic to be built:

- `xor` specifies to build an XOR-based spreader network in addition to the broadcast-based decompression logic
- `broadcast` specifies to build a broadcast-based decompression logic (simple scan fanout).

`-file` *`file`*

Specifies the filename where the compression macro RTL will be written. If not specified, the RTL will be written to `stdout`.

`-info_file` *`file`*

Specifies a file containing more detailed information about the compression macro.

`-jtag_control`

Specifies to include a JTAG-controlled test data register (TDR) which generates compression test signals to configure the compression testmode.

`-low_pin_compression`

Enables low pin count compression.

`-mask {wide0| wide1 | wide2}`

Inserts scan channel masking logic of the specified type.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The masking types that can be used depend on the compressor type specified with the `-compressor` option.

`-mask_sharing_ratio` *`integer`*

Specifies the number of internal scan channels sharing a mask register. The specified integer may not exceed the value specified for the compression ratio.

Note: This option is only valid with `wide1` and `wide2` masking.

`-no_fullscan_muxing`

Specifies to exclude additional muxing logic to the compression macro. By default, additional muxing is added to the compression macro to concatenate the compressed scan channels into uncompressed fullscan chains. If such muxing exists outside the compression macro, specify this option to exclude this logic from the compression macro.

`-remove_cancelling_terms_for_hierarchical_flow`

Generates a macro which prevents cancellation of compression channels within higher-level compression macros. Each scan channel is observed at an odd number of scan outputs.

`-scan_in` *`integer`*, `-scan_out` *`integer`*

Specify the width of the `RSI_SI` and `DSO_SO` ports (the number of scan data inputs and scan data outputs to the compression macro) respectively.

These options apply for MxN compression in any of the following cases:

- M<N
- M<N-1 and wide1 masking is used: one of the scan data inputs can be shared with the CME pin
- M<N-2 and wide2 masking is used: two of the scan data inputs can be shared with the CME0 and CME1 pins
- M>N

`-scan_in_pipeline_depth` *`integer`*

Specifies the number of pipeline stages required at the scan data input side.

`-scan_out_pipeline_depth` *`integer`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the number of pipeline stages required at the scan data output side.

`-separate_mask_ports`

Creates separate `MASK_IN` and `MASK_OUT` ports that are used for block level compression processing in the hierarchical compression flow.

The number of `MASK_IN/MASK_OUT` ports that is added, is the same number as the number of scan data input ports.

`-serial_misr_read` Specifies to include support for reading MISR bits serially through the scan data pins.

`-shared_scan_in_pins` *`string`*

Specifies the pins that can be shared with the scan data input pins. Separate the pin names using one or more blanks.

- For `-mask wide1`, specify *`CME`*
- For `-mask wide2`, you can specify the following values: `CME0,CME1,` or `CME0 CME1`

This option enables the use of an asymmetrical compression macro.

This option will be ignored and a warning will be given if you also specified the `-scan_in` and `-scan_out` options and the value for `scan_in` is larger than the value for `scan_out`.

`-smartscan_no_update_stage`

Prevents the insertion of update registers between the deserializer and the decompressor. In this case, lockup latches are inserted between the deserializer flops and the decompressor.

You cannot specify this option when you have set the `-smartscan_pulse_width_multiplier` option to either `2` or `4`.

By default, the tool inserts update registers between the deserializer and the decompressor.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-smartscan_pulse_width_multiplier {1|2|4}`

Determines whether to add clock divider logic to widen the clock pulse going to the scan chains. You can specify the following values:

- 1—no logic added
- 2—increases the scan path through the SmartScan clock controller with 1 bit
- 4—increases the scan path through the SmartScan clock controller with 2 bits

Default: `1`

`-smartscan_ratio` *`integer`*

Specifies the number of parallel scan data input pins that correspond to a single serial scan data input pin. The number of defined (fullscan) chains must be an integral multiple of the specified smartscan ratio.

`-smartscan_serial_only`

Specifies to only insert the smartscan serial-only interface. The number of deserializer (and serializer) registers will match the number of the defined chains. When building the model for Modus you will need to add the pseudo pins for the parallel interface.

`-sub_chains` *`integer`*

Specifies the number of compressed scan channels that exist in the design or that you will build.

Examples

- The following command writes an XOR-based compression macro without masking to the file `xor1.v`.

```
legacy_genus:/> write_compression_macro -compressor xor -chains 8 \
-sub_chains 88 -file xor1.v
Checking out license 'Encounter_Test_Architect'... (1 seconds elapsed)
...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command writes an XOR-based compression macro with masking logic of type `wide1`. The `-no_fullscan_muxing` option is specified so the logic to concatenate the compressed `sub_chains` into fullscan chains will be excluded.

  Note: Since the `-no_fullscan_muxing` option is specified, the number of sub_chains is no longer required to be evenly divisible by the number of chains.

  ```
  legacy_genus:/> write_compression_macro -compressor xor -mask wide1 -chains 8 \
  -sub_chains 85 -no_fullscan_muxing -file xor2.v
  Checking out license 'Encounter_Test_Architect'... (1 seconds elapsed)....
  ```

- The following command writes an MISR-based compression macro with masking logic of type `wide1`, with decompression logic of type `xor`. The compressor type is `hybrid` which means the MISR can be bypassed resulting in XOR compression.

  ```
  legacy_genus:/> write_compression_macro -compressor hybrid -decompressor xor \
  -mask wide1 -chains 16 -sub_chains 512 -file hybrid1.v
  Checking out license 'Encounter_Test_Architect'... (2 seconds elapsed)
  ...
  ```

- The following command writes a MISR-based compression macro with masking logic of type `wide2`. The `-info_file` option is also specified.

  ```
  legacy_genus:/> write_compression_macro -compressor opmisr -mask wide2 \
   -chains 20 -sub_chains 500 -file opmisr1.v -info_file opmisr1.info
  ....
  ```

- The following command generates an asymmetric pipelined XOR-based compression macro, with masking logic of type `wide2`, with five top-level chains, of which two are shared with the mask control signals, with separate mask ports, with additional logic to bypass the block for compression inserted at the block level, and with two pipeline stages at the scan data output side.

  ```
  write_compression_macro -chains 5 -sub_chains 9 -asymmetric_scan_in 3 \
  -shared_scan_in_pins "CME0 CME1" -mask wide2 -separate_mask_ports \
  -block_select -scan_out_pipeline_depth 2 -file my_comp_macro.v
  ```

- The following command generates a pipelined XOR-based compression macro, with masking logic of type `wide1`, with two top-level chains, with one pipeline stage at the scan data input side, six pipeline stages at the scan data output side, and without additional muxing logic.

  ```
  write_compression_macro -chains 2 -sub_chains 5 -mask wide1 \
  -scan_in_pipeline_depth 1 -scan_out_pipeline_depth 6 -no_fullscan_muxing
  ```

- The following command generates an asymmetric SmartScan-based compression macro with both parallel and serial interface, with masking logic of type `wide1`, with eight fullscan chains, 32 compression channels and a SmartScan ratio of 4.

  ```
  write_compression_macro -chains 8 -sub_chains 32 -mask wide1 \
  -compressor smartscan_xor -decompressor xor  -smartscan_ratio 4
  ```

- The following command generates an asymmetric SmartScan-based compression macro with serial interface only, with masking logic of type `wide1`, with two fullscan chains, 32 compression channels and a SmartScan ratio of 4.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
write_compression_macro -chains 2 -sub_chains 32 -mask wide1 \
-compressor smartscan_xor -decompressor xor  -smartscan_ratio 4 \
-smartscan_serial_only
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_abstract_model

```
write_dft_abstract_model [-ctl]
     [-include_compression_information]
     [-dft_configuration_mode dft_config_mode]
     [-include_opcg_domain_information]
     [-write_as_libcell]
     [design] [> file]
```

Writes a scan abstract model for all the top-level scan chains configured in the design.

Besides the command options, following root attributes also affect the information written to the abstract models:

- The `dft_include_controllable_pins_in_abstract_model` attribute allows for feedthrough pin connections to be written using the `dft_controllable` construct to the native abstract model, and using the `IsConnected` construct to the CTL model.
- The `dft_include_test_signal_outputs_in_abstract_model` attribute allows output signals whose values are constant in test setup, and output signals assigned to tied constant values, to be written as test mode signals in the native abstract model, and as constants in a CTL abstract model.

Note: Currently, this command is not supported for the clocked LSSD scan style with the `-ctl` option.

Options and Arguments

`-ctl`
: Writes out a scan abstract model in the Core Test Language (CTL) format (IEEE format `P1450.6`).

  If you omit this option, the scan abstract model is written in native RC format that consists of a list of `define_dft_abstract_segment` commands, one per top-level scan chain.

`design`
: Specifies the design for which to write out the scan abstract model of the scan chains.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_configuration_mode` *`dft_configuration_mode`*
: Writes scan chain information related to the specified scan mode name.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

file
: Specifies the file to which the output must be written.

You can write out the CTL file in compressed format by specifying a file name with the `.gz` extension.

Default: output is written to the screen

`-include_compression_information`
: Adds the required compression information created by the `compress_block_level_chains` command. This information is used at the next level of integration in the hierarchical compression flow.

`-include_opcg_domain_information`
: Adds the required OPCG clock domain information created during block-level processing in a domain-blocking flow.This information is applied to the block at the next level of integration and used when inserting OPCG logic with domain blocking at the top-level design.

`-write_as_libcell`
: Writes out the abstract information as a libcell.

Examples

- In the following example, the different active edges of the different test clocks in the same test clock domain are allowed to be mixed on the same scan chains. Following shows the configuration result and the scan abstract models for the scan chains:

```
legacy_genus:/> connect_scan_chains
    Configuring 1 chains for 27 scan f/f
  Configured 1 chains for Domain: 'clkAll', edge: 'mixed'
         AutoChain_1 (DFT_sdi_1 -> DFT_sdo_1) has 27 registers; Domain:clkAll,
edge: mixed
  Processing 1 scan chains in 'muxed_scan' style.
  Using default shift enable signal 'SE': '/designs/test/ports_in/SE' active
high.
    Connecting scan chain 'AutoChain_1' with 27 flip-flops.
Mapping DFT logic introduced by scan chain connection...
Mapping DFT logic done.
Reporting 1 scan chain

Chain 1: AutoChain_1
  scan_in:      DFT_sdi_1
  scan_out:     DFT_sdo_1
  shift_enable: SE (active high)
  clock_domain: clkAll (edge: mixed)
  length: 27
    bit 1       out1_reg_4 <test_clk1/fall>
    ...
    bit 5       out1_reg_8 <test_clk1/fall>
    llatch 5    DFT_lockup_g1
    bit 6       out2_reg_4 <test_clk2/fall>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
    ...
    bit 10      out2_reg_8 <test_clk2/fall>
    llatch 10   DFT_lockup_g348
    bit 11      out3_reg_4 <test_clk3/fall>
    ...
    bit 18      out1_reg_2 <test_clk1/rise>
    bit 19      out1_reg_3 <test_clk1/rise>
    llatch 19   DFT_lockup_g349
    bit 20      out2_reg_0 <test_clk2/rise>
    ...
    bit 23      out2_reg_3 <test_clk2/rise>
    llatch 23   DFT_lockup_g350
    bit 24      out3_reg_0 <test_clk3/rise>
    ...
    bit 27      out3_reg_3 <test_clk3/rise>
------------------------

legacy_genus:/> write_dft_abstract_model
scan style is muxed_scan

# writing abstract model for 1 scan chain

  define_dft abstract_segment -module test \
    -name test_AutoChain_1 \
    -sdi DFT_sdi_1 -sdo DFT_sdo_1 \
    -shift_enable_port SE -active high \
    -clock_port clk1 -fall \
    -tail_clock_port clk3 -tail_edge_rise \
    -length 27
```

To avoid naming collisions when reading in a scan abstract model, the segment names are prefixed with the module name.

Related Information

Affected by these commands:	compress_block_level_chains on page 878
	connect_scan_chains on page 906
Affected by these attributes:	Actual Scan Chain attributes
	dft_include_controllable_pins_in_abstract_model
	dft_include_test_signal_outputs_in_abstract_model
Related command:	read_dft_abstract_model on page 1113
	write_hdl on page 404 (`-abstract`)

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_compression_test_points

```
write_dft_compression_test_points
     [design] [-library string]
     [-directory string] [-run_from_directory]
     [-compression_testpoints_options string]
     [-build_model_options string]
     [-build_faultmodel_options string]
     [-build_testmode_options string]
     [-verify_test_structures_options string]
     [-min_slack_for_no_tp_file integer]
     [-force] [-continue_with_severe]
```

Writes out the necessary files and the template run scripts to run scripts to run Compression Testpoint Analysis using the Modus software.

This command generates the following files:

- `test.exclude`—A file listing objects to be excluded from the test point analysis in an assumed scan mode.
- `test.modedef`—A mode definition file, a text file that describes the test mode in assumed scan mode
- *`topmodulename`*`.ASSUMED.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock) and their test function used to build the testmode before actual scan chains exist in the design
- *`topmodulename`*`.FULLSCAN.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock and scan data IOs) and their test function used to build the testmode when actual scan chains exist in the design
- *`topmodulename`*`.noTpFile`—A file listing objects that have been constrained, thus preventing test point insertion on them.
- *`topmodulename`*`.test_netlist.v`—A netlist for Modus.
- `runmodus.compression_testpoints.tcl`—A script file to run compression test point analysis in Modus.

Note: Some files can be customized according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-build_faultmodel_options "-option1 value -option2 value"`

Specifies a string containing the extra options to build a fault model.

`-build_model_options "-option1 value -option2 value"`

Specifies extra options to apply when building the model.

`-build_testmode_options "-option1 value -option2 value"`

Specifies extra options to apply when building the test mode.

`-compression_testpoints_options "-option1 value -option2 value"`

Specifies the extra options to run Compression Testpoints Analysis in a string.

`-continue_with_severe`

Allows the continuation of the test script even when severe warnings occur during execution of the commands in the script.

design

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *string*

Specifies the directory to which the output files must be written.

Default: *current_working_directory*/`test_scripts`

`-force`

Specifies to continue even if the JTAG macro used to enable the JTAG-controlled compression macro is not found.

`-library` *string*

Specifies the list of Verilog structural library files.

The Verilog libraries required to run ATPG of the generated vectors must be provided using the `-library` option of the `write_dft_compression_test_points` command. These libraries must be in a structural format. The files can be specified separately on the `write_dft_compression_test_points` command line or can be referenced using an *include* file. Directories of Verilog files can also be specified but they cannot be referenced in the include file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

For example, if the Verilog files required to run ATPG are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_dft_compression_test_points` can be used in either of the following ways:

1. If specifying files separately on the command line:

   ```
   write_dft_compression_test_points -library
   "./padcells.v ./stdcells.v \
    ./memories ./ip_blocks" ...
   ```

2. If using an include file. Create an include file named `include_libraries.v` containing:

   ```
   `include "./padcells.v"
   `include "./stdcells.v"
   ```

   And then specify the following:

   ```
   write_dft_compression_test_points -library
   "include_libraries.v \ ./memories  ./ip_blocks" ...
   ```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-min_slack_for_no_tp_file` *`integer`*

Prevents test point insertion on a pin, if the pin does not have the specified minimum slack.

If the slack is lower, and the `dft_generate_atpg_no_testpoint_file` root attribute is set to `true`, the tool adds the pin to a file that prevents Modus from inserting a test point on such pins.

`-rrfa_options "`*`-option1 value -option2 value`*`"`

Specifies the extra options to run RRFA-based testability analysis in a string.

`-run_from_directory`

Allows to run the `runmodus.compression_testpoints.tcl` script from the Modus working directory.

Default: Directory where scripts were written from, that is, directory where Genus was run.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-verify_test_structures_options "-option1 value -option2 value"`

Specifies extra options to apply when performing test structure verification.

Examples

- The following command generates the files to run a compression-based testability analysis.

```
write_dft_compression_test_points -directory comp -library $sim/mylib.v
```

Related Information

Affected by this attribute	dft_generate_atpg_no_testpoint_file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_deterministic_test_points

```
write_dft_deterministic_test_points
     [-library string]
     [-directory string] [-run_from_directory]
     [-build_model_options string]
     [-build_faultmodel_options string]
     [-build_testmode_options string]
     [-atpg_options string] [-dfa_options string]
     [-include_redundant_faults]
     [-verify_test_structures_options string]
     [-min_slack_for_no_tp_file integer]
     [-continue_with_severe] [-unix_shell]
     [design]
```

Writes out the necessary files and scripts to run Deterministic Fault Analysis (DFA) using the Modus software. The template script will only be written if actual scan chains exist in the design. DFA analysis is not supported in `ASSUMED` scan mode.

This command generates the following files:

- *topmodulename*`.FULLSCAN.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock and scan data IOs) and their test function used to build the testmode when actual scan chains exist in the design
- `runmodus.dfa.tcl` —A template script file to run the requested deterministic fault analysis.
- *topmodulename*`.test_netlist.v`—A netlist for Modus.
- *topmodulename*`.noTpFile`—A file listing objects that are constrained, thus preventing test point insertion on them.

Note: Some files can be customized according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-atpg_options "-option1 value -option2 value"`

Specifies extra ATPG analysis options.

`-build_faultmodel_options "-option1 value -option2 value"`

Specifies a string containing the extra options to build a fault model.

`-build_model_options "-option1 value -option2 value"`

Specifies extra options to apply when building the model.

`-build_testmode_options "-option1 value -option2 value"`

Specifies extra options to apply when building the test mode.

`-continue_with_severe`

Allows the continuation of the test script even when severe warnings occur during execution of the commands in the script.

design

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dfa_options "-option1 value -option2 value"`

Specifies extra options for deterministic fault analysis.

`-directory string`

Specifies the directory to which the output files must be written.

Default: `current_working_directory/test_scripts`

`-include_redundant_faults`

Specifies to include the redundant faults during DFA analysis.

`-library string`

Specifies the list of Verilog structural library files.

The Verilog libraries required to run ATPG must be provided using the `-library` option of the `write_dft_dfa` command. These libraries must be in a structural format. The files can be specified separately on the `write_dft_dfa` command line or can be referenced using an *include* file. Directories of Verilog files can also be specified but they cannot be referenced in the include file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

For example, if the Verilog files required to run ATPG are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_dft_dfa` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_dft_dfa -library "./padcells.v ./stdcells.v  ./
memories ./ip_blocks" ...
```

2. If using an include file. Create an include file named `include_libraries.v` containing:

```
`include "./padcells.v"
`include "./stdcells.v"
```

And then specify the following:

```
write_dft_dfa -library "include_libraries.v ./
memories ./ip_blocks" ...
```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-min_slack_for_no_tp_file` *`integer`*

Prevents test point insertion on a pin, if the pin does not have the specified minimum slack.

If the slack is lower, and the `dft_generate_atpg_no_testpoint_file` root attribute is set to `true`, the tool adds the pin to a file that prevents Modus from inserting a test point on such pins.

`-run_from_directory`

Allows to run the Deterministic Fault Analysis from the working directory specified with the `-directory` option.

Default: current working directory of Genus.

`-unix_shell` Generates the files in shell format for use with Encounter Test.

`-verify_test_structures_options "`*`-option1 value -option2 value`*`"`

Specifies extra options to apply when performing test structure verification.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command generates the files to run deterministic fault analysis.

```
write_dft_deterministic_test_points -directory dfa -library $sim/mylib.v
```

Related Information

Related command:	add_analyzed_test_points on page 848
Affected by this attribute	dft_generate_atpg_no_testpoint_file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_lbist_test_points

```
write_dft_lbist_test_points
     [-lbist_testpoint_options string]
     [-build_model_options string]
     [-build_faultmodel_options string]
     [-build_testmode_options string]
     [-verify_test_structures_options string]
     [-library string]
     [-directory string] [-run_from_directory]
     [-min_slack_for_no_tp_file integer]
     [-force] [-continue_with_severe]
     [-unix_shell] [design]
```

Writes out the necessary files and the template run scripts to run a Random Resistant Fault Analysis (RRFA) based testability analysis using the Modus software.

This command generates the following files:

- `test.exclude`—A file listing objects to be excluded from the **ATPG** analysis in an `ASSUMED` scan mode.
- `test.modedef`—A mode definition file, a text file that describes the test mode in `ASSUMED` scan mode.
- *`topmodulename`*`.ASSUMED.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock) and their test function used to build the testmode before actual scan chains exist in the design
- *`topmodulename`*`.FULLSCAN.pinassign`—A pin-assignment file that captures the top-level scan-related signals (shift-enable, test-mode, test-clock and scan data IOs) and their test function used to build the testmode when actual scan chains exist in the design
- *`topmodulename`*`.noTpFile`—A no test point listing objects that are constrained, thus preventing test point identification on them.
- *`topmodulename`*`.test_netlist.v`—A Verilog netlist for use with Modus.
- `runmodus.rrfa.tcl`—A script to run random resistant fault analysis in Modus.
- `runet.rrfa`—A script to run random resistant fault analysis in Encounter Test.

Note: Some files can be customized according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-build_faultmodel_options "-option1 value -option2 value"`

Specifies a string containing the extra options to build a fault model.

`-build_model_options "option1 value -option2 value"`

Specifies extra options to apply when building the model.

`-build_testmode_options "-option1 value -option2 value"`

Specifies extra options to apply when building the test mode.

`-continue_with_severe`

Allows the continuation of the test script even when severe warnings occur during execution of the commands in the script.

design

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *string*

Specifies the directory to which the output files must be written.

Default: *current_working_directory*/`test_scripts`

`-force`

Specifies to continue even when the DFT logic appears to be incomplete.

`-lbist_testpoints_options "-option1 value -option2 value"`

Specifies the extra options to run Lbist testpoint analysis.

`-library` *string*

Specifies the list of Verilog structural library files.

The Verilog libraries required to run ATPG must be provided using the `-library` option of the `write_dft_lbist_test_points` command. These libraries must be in a structural format. The files can be specified separately on the `write_dft_lbist_test_points` command line or can be referenced using an *include* file. Directories of Verilog files can also be specified but they cannot be referenced in the include file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

For example, if the Verilog files required to run ATPG are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_dft_lbist_test_points` can be used in either of the following ways:

1. If specifying files separately on the command line:

   ```
   write_dft_lbist_test_points -library "./padcells.v
   ./stdcells.v ./memories ./ip_blocks" ...
   ```

2. If using an include file. Create an include file named `include_libraries.v` containing:

   ```
   `include "./padcells.v"
   `include "./stdcells.v"
   ```

   And then specify the following:

   ```
   write_dft_lbist_test_points -library
   "include_libraries.v ./memories  ./ip_blocks" ...
   ```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-min_slack_for_no_tp_file` *`integer`*

Prevents test point insertion on a pin, if the pin does not have the specified minimum slack.

If the slack is lower, and the `dft_generate_atpg_no_testpoint_file` root attribute is set to `true`, the tool adds the pin to a file that prevents Modus from inserting a test point on such pins.

`-run_from_directory`

Allows to run either Automatic Test Pattern Generator (ATPG) or Random Resistance Fault Analysis (RRFA) from the working directory specified with the `-directory` option.

Default: parent directory of the specified work directory.

`-unix_shell` Generates the files for use with Encounter Test.

`-verify_test_structures_options "`*`-option1 value -option2 value`*`"`

Specifies extra options to apply when performing test structure verification.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command generates the files to run an LBIST-based testability analysis.

```
write_dft_lbist_test_points -directory lbist -library $sim/mylib.v
```

Related Information

Affected by this attribute	dft_generate_atpg_no_testpoint_file

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_opcg_domain_macro_rtl

```
write_dft_opcg_domain_macro_rtl
     [-name opcg_domain_module]
     [-max_num_pulses integer]
     [-counter_length integer]
     [-domain_blocking] [-los]
     [-file file]
```

Writes out the RTL of an OPCG domain. By default, Genus creates a gate level version of the OPCG Macro and instantiates it in the design. This is the optimized during subsequent synthesis optimizations. Some users may want on RTL version of this macro to use in RTL level simulations. This would typically be done for performance reasons as RTL simulation is much faster than gate level. The simulation of a gate level OPCG macro with the rest of the RTL can significantly degrade simulation performance.

Options and Arguments

`-counter_length` *`integer`*

Specifies the number of bits in the down counter register.

Specify a number between 4 and 7, or specify 0 if no counter should be inserted.

`-domain_blocking` Implements OPCG domain blocking.

`-file` *`file`* Specifies the file to which to write the out the OPCG domain macro,

Default: `stdout`

`-los` Implements OPCG Launch-Off-Shift (LOS).

`-max_number_pulses` *`integer`*

Specifies the number of high speed pulses the domain macro should be able to generate. The maximum number you can specify is 8.

Default: `2`

`-name` *`name`* Specifies the name of the OPCG domain module.

Related Information

RTL OPCG section in *Genus Design for Test Guide for Legacy UI*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_opcg_trigger_macro_rtl

```
write_dft_opcg_trigger_macro_rtl
     [-name opcg_trigger_module]
     [-delay_cycles integer]
     [-file file]
```

Writes out the RTL of an OPCG trigger macro.

Options and Arguments

`-delay_cycles` *`integer`*

Specifies the number of extra cycles to be used to delay the trigger signal inside the trigger macro.

Default: `stdout`

`-file` *`file`*

Specifies the file to which to write the out the OPCG trigger macro,

Default: `0`

`-name` *`name`*

Specifies the name of the `OPCG` domain module.

Related Information

RTL OPCG section in *Genus Design for Test Guide for Legacy UI*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_dft_rtl_model

```
write_dft_rtl_model
     -directory string [design]
```

Writes an RTL model of the design in Verilog if the DFT RTL insertion flow was enabled and DFT insertion commands supporting the RTL update flow were run. DFT RTL insertion flow is enabled using attribute `dft_rtl_insertion`. The commands that support RTL insertion of DFT logic are `insert_dft jtag_macro`, `insert_dft pmbist` and `insert_dft hard_repair`.

This command analyzes and writes all the design files. The written design files contain three kinds of files.

1. *design_name*_eq.v file - This is a new file. This file contains the definition of DFT modules which were added to the user design.
2. Modified RTL source files - These are the files which are modified by this command. The modifications include the instantiation of DFT logic and their connections to existing design modules. The command prints the list of modified files on the standard output.
3. Original design files - These are the files which are not modified by this command.

The directory structure of the original design source is kept intact while writing the design files under user supplied location. Pure Verilog designs and mixed designs are supported with this command as long as the modifications are required for Verilog modules only.

Note: The attribute `dft_rtl_insertion` must be set prior to running `elaborate` command on the design so that all the design changes can be tracked and applied properly while writing the DFT inserted RTL design.

The DFT inserted RTL design now becomes the new golden design which can be used for synthesis. While treating DFT inserted RTL as new golden design, the attribute `dft_rtl_insertion` must be disabled so that synthesis can be performed.

Options and Arguments

design	Specifies the design for which to write out the Modus input files. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
-directory *string*	Specifies the directory to which the updated RTL files must be written.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*

- Using Modus to Automatically Select and Insert Test Points
- Exporting the Design for Test Point Selection

Affected by this attribute dft_rtl_insertion

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_atpg

```
write_et_atpg
     [-ncsim_library string [-library string]]
     [-directory string] [-run_from_directory]
     [ -configuration_mode_order dft_configuration_mode_list]
     [ -delay ]
     { [-opcg_mode opcg_mode]
     | [-compression]
       [-dft_configuration_mode dft_config_mode]...
       [-build_faultmodel_options string]
       [-build_model_options string]
       [-build_testmode_options string]
       [-atpg_options string]
       [-verify_test_structures_options string] }
     [-force] [-continue_with_severe] [-serial]
     [-hier_test_core] [-hier_test_top] [-unix_shell] [design]
```

Writes out the necessary files and the template run scripts to run Automatic Test Pattern Generator (ATPG) using the Modus software.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

The generated scripts depend on the setting of the `dft_true_time_flow` root attribute.

he generated scripts depend on the setting of the `dft_true_time_flow` root attribute. There is one additional file called `tt_setup` that is written out when you set `dft_true_time_flow` attribute to true. Use the Modus `true_time` command to regenerate the `runmodus.atpg.tcl` script after making modifications to the `tt_setup` file.

This command generates the following files:

- *topmodulename*`.et_netlist.v`—A netlist for Modus
- `runmodus.atpg.tcl` —A template script file to run the requested tasks such as pattern generation or testability analysis
- *topmodulename*`.`*TESTMODE*`.pinassign`—A pin-assignment file that captures the top-level scan-related signals and their applicable test function used to build and validate the specific testmode. For example the test signals for FULLSCAN testmode include shift-enable, test-mode, test-clock and scan data IOs.For OPCG- based testmodes, this file also includes the OPCG test signals with their appropriate test functions and the oscillator and domain related information.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- *topmodulename*.*TESTMODE_OPCGModeName*.seqdef—A sequence definition file generated when OPCG logic is inserted in the design and the command is run with the -delay option. For the true_time flow, add the -opcg_mode option as well. This file is used to initialize the PLL, establish the OPCG mode, and validate their specific test mode.
- run_*TESTMODE*_sim[_*OPCGModeName*] — An Xcelium simulator run file used to simulate the test patterns created by Modus for the given test mode on the design containing the applicable test logic.
- run_*TESTMODE*_sim[_*OPCGModeName*]_sdf — An Xcelium simulator run file used to simulate the test patterns created by Modus with SDF back annotated for the given test mode on the design containing the applicable test logic.

When using the -unix_shell option, a similar set of files as listed above are generated for use with Encounter Test. The name of the netlist file is *topmodulename*.et_netlist.v. The Encounter Test run script is named runet.atpg. In addition, an et_checks.sh file is generated.

Options and Arguments

-atpg_options {*-option1 value -option2 value*}

Specifies extra ATPG analysis options to include for create_logic_tests commands in the runmodus.atpg.tcl script.

Specify the extra options as "*option1*=*value option2*=*value*" when using -unix_shell argument as part of the command.

Note: This option does not apply to the true time flow.

-build_faultmodel_options {*-option1 value -option2 value*}

Specifies a string containing the extra options to build a fault model in the runmodus.atpg.tcl script.

Specify the extra options as "*option1*=*value option2*=*value*" when using -unix_shell argument as part of the command.

Note: For more information on these options, refer to the build_faultmodel command in the Modus documentation.

Note: This option does not apply to the true time flow.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-build_model_options {-option1 value -option2 value}`

Specifies extra options to apply when building the model in the `runmodus.atpg.tcl` script.

Specify the extra options as "*option1=value option2=value*" when using -`unix_shell` argument as part of the command.

Note: For more information on these options, refer to the `build_model` command in the Modus documentation.

Note: This option does not apply to the true time flow.

`-build_testmode_options {-option1 value -option2 value}`

Specifies extra options to apply when building the test mode.

Specify the extra options as "*option1=value option2=value*" when using -`unix_shell` argument as part of the command.

Note: For more information on these options, refer to the `build_testmode` command in the Modus documentation.

Note: This option does not apply to the true time flow.

`-cfg_mode_order dft_configuration_mode...`

Specifies to write Modus script files for a compression mode. Valid compression mode names are:

`COMRESSION`, `COMRESSION_DECOMP`, `OPMISRPLUS`, `OPMISRPLUS_DECOMP`, `FULLSCAN`

Note: If specified, the `FULLSCAN` compression mode must be specified last.

- In the default flow, you cannot combine this option with the `-delay` option.
- In the true time flow, you can combine this option with the `-delay` option, but you can only specify two configuration modes because the true time flow can only handle two modes.

`-compression`

Generates additional files to run ATPG and to verify compression logic.

`-continue_with_severe`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Allows the continuation of the Modus script even when severe warnings occur during execution of the commands in the script.

`-delay`

Specifies to generate additional files for Modus to run delay tests. In the default flow, the design contains OPCG logic.

- In the default flow, you cannot combine this option with the `-cfg_mode_order` option.
- In the true time flow, you can combine this option with the `-cfg_mode_order` option. In this flow, this option specifies to write out delay tests even when no OPCG logic was inserted.

design

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_cfg_mode` *dft_configuration_mode...*

Writes scan chain information related to the specified scan mode name(s).

Note: This option does not apply to the true time flow.

`-directory` *string*

Specifies the directory to which the output files must be written.

Default: *current_synthesis_run_directory/*`test_scripts`

`-force`

Specifies to skip certain checks to validate the design and test setup are complete and ready for ATPG. Allows the `write_dft_atpg` command to continue even if the DFT logic appears to be incomplete.

`-hier_test_core`

Instructs to write out the files needed to generate migratable patterns for a hierarchical test core flow.

The design must have been taken through the "Hierarchical Test Flow: Preparing a Core" flow.

`-hier_test_top`

Instructs to write out the files to migrate the core patterns and generate SoC level patterns for a hierarchical test top flow.

`-library` *string*

Specifies the list of Verilog structural library files required to run Modus ATPG. For mapped design, this option is required.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can specify the files explicitly or you can specify an *include* file that lists the files. You can also specify directories of Verilog files but you cannot reference directories in an include file.

For example, if the Verilog files required to run ATPG are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_dft_atpg` can be used in either of the following ways:

1. If specifying files separately on the command line:

   ```
   write_dft_atpg -library "./padcells.v ./stdcells.v \
    ./memories ./ip_blocks" ...
   ```

2. If using an include file. Create an include file named `include_libraries.v` containing:

   ```
   `include "./padcells.v"
   `include "./stdcells.v"
   ```

   And then specify the following:

   ```
   write_dft_atpg -library "include_libraries.v ./
   memories \
    ./ip_blocks" ...
   ```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-ncsim_library` *`string`*

Specifies the list of library files required for the Xcelium simulation of the generated vectors. If this option is not used, then the list of libraries specified with `-library` option are used.

For more information on how to specify the list of files, refer to the `-library` option.

`-opcg_mode` *`opcg_mode`*

Specifies the OPCG mode for which the delay tests must be generated.

Note: This option only applies to the true time flow.

`-run_from_directory`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Allows to run the Automatic Test Pattern Generator (ATPG) from the working directory specified with the `-directory` option.

Default: directory where scripts were written from, that is, directory where Genus was run.

`-serial`
Generates additional files for Modus to verify serial testmodes: `SERIAL_INTEST`, `SERIAL_EXTEST`, and `SERIAL_BYPASS` for use with hierarchical test/ATPG flows.

`-unix_shell`
Generates the run script in shell format, for example, *`command "option1=value option2=value"`*

`-verify_test_structures_options {-option1 value -option2 value}`

Specifies extra options to apply when performing test structure verification for Modus.

Specify the extra options as "*`option1=value option2=value`*" when using `-unix_shell` argument as part of the command.

Note: This option does not apply to the true time flow.

Examples

- The following command generates the files to run an ATPG-based testability analysis.

```
write_et_atpg -directory atpg -library $sim/mylib.v
Examining the atpg directory that was generated shows the following files:
legacy_genus:/>  shell ls atpg
run_compression_decomp_sim
run_compression_sim
runet.atpg
run_fullscan_sim
test.COMPRESSION_DECOMP.pinassign
test.COMPRESSION.pinassign
test.et_netlist.v
test.FULLSCAN.pinassign
test.rc_netlist.v
```

- The following command uses the `-configuration_mode_order` option to generate the files to run an ATPG-based testability analysis first using the `OPMISRPLUS_DECOMP` compression mode and then with the `FULLSCAN` mode.

```
write_et_atpg -configuration_mode_order {OPMISRPLUS_DECOMP FULLSCAN} \
 -directory rc_et
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*:

- Using Modus to Analyze Testability
- Writing the Scripts and Setup Files to Perform ATPG in “Exporting the Design”
- Defining the Oscillator Sources in “Inserting On-Product Clock Generation Logic”
- Generate Files for ATPG and Simulation in “Inserting On-Product Clock Generation Logic”
- Compression Logic Verification and Test Vector Generation Using Modus in “Inserting Scan Chain Compression Logic”
- Using Modus to Perform a Deterministic Fault Analysis on a Scan Connected Netlist
- Hierarchical Test Flow: Preparing a Core in “Herarchical Test”
- Hierarchical Test Flow: Integrating the Top Level in “Hierarchical Test”
- ICL/PDL for Core and Top Level in “Hierarchical Test”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_bsv

```
write_et_bsv -library string
     [-bsdl file [-bsdl_package_path string]
          [-bsdl_package_name files]]
     [-build_model_options string]
     [-directory string] [-run_from_directory]
     [-continue_with_severe] [-unix_shell] [design]
```

Writes out the necessary files and the template run scripts to run boundary scan verification.

This command generates the following files:

- *topmodulename*.bsdl—A BSDL file
- *topmodulename*.test_netlist.v—A netlist for Modus
- runmodus.bsv.tcl—A Modus run file to run Boundary Scan Verification (BSV).

Note: Some files can be customized according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

-bsdl *file*

Specifies the name of the BSDL file to be used for the boundary scan verification.

If you omit this option but you specified the -bsv option, this command will automatically run the write_bsdl command to generate the BSDL file.

Important

You must use this option if you inserted custom boundary cells in the design. Additionally, the BSDL file should be written using the write_bsdl command with the -bsdl_package_name option to list the custom package file to be used during boundary scan verification.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-bsdl_package_name` *`files`*

Specifies a package file or a comma-separated list of package files that describe the custom boundary cells used in the design.

Note: This option is only required if you used custom boundary cells in the design. This option cannot be specified without the `-bsdl` option.

`bsdl_package_path` *`string`*

Specifies the UNIX directory or a comma-separated list of directories that indicate(s) where to find the package file(s).

You can use dot (.) to refer to the current working directory.

Note: This option is only required if you used custom boundary cells in the design. This option cannot be specified without the `-bsdl` option.

`-build_model_options {`*`-option1 value -option2 value`*`}`

Specifies extra options to apply when building the model.

`-continue_with_severe`

Allows the continuation of the Modus script even when severe warnings occur during execution of the commands in the script.

`design`

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *`string`*

Specifies the directory to which the output files must be written.

Default: *`current_working_directory/`*`test_scripts`

`-library` *`string`*

Specifies the list of Verilog structural library files.

The Verilog libraries required to run ATPG and Xcelium simulation of the generated vectors must be provided using the `-library` option of the `write_et_bsv` command. These libraries must be in a structural format. The files can be specified separately on the `write_et_bsv` command line or can be referenced using an *include* file. Directories of Verilog files can also be specified but they cannot be referenced in the include file.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

For example, if the Verilog files required to run ATPG and Xcelium simulation are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_et_bsv` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_et_bsv -library "./padcells.v ./stdcells.v \
 ./memories ./ip_blocks" ...
```

2. If using an include file. Create an include file named `include_libraries.v` containing:

```
`include "./padcells.v"
`include "./stdcells.v"
```

And then specify the following:

```
write_et_bsv \
 -library "include_libraries.v ./memories \
 ./ip_blocks" ...
```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-run_from_directory`

Allows to run boundary scan verification from the working directory specified with the `-directory` option.

Default: parent directory of the specified work directory.

`-unix_shell` Generates the files for use with Encounter Test.

Examples

- The following command generates the files to run an ATPG-based testability analysis.

```
write_et_bsv -directory bsv -library $sim/mylib.v
```

Examining the `atpg` directory that was generated shows the following files:

```
legacy_genus:/>  shell ls bsv
topmodulename.bsdl
runet.bsv
test.et_netlist.v
test.rc_netlist.v
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Generating Script for Boundary Scan Verification in *Genus Design for Test Guide for Legacy UI*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_lbist

```
write_et_lbist
     [-build_model_options {-option1 value -option2 value}]
     [-build_testmode_options {-option1 value -option2 value}]
     [-build_faultmodel_options {-option1 value -option2 value}]
     [-verify_test_structures_options {-option1 value -option2 value}]
     [-library string]  [-ncsim_library string]
     [-directory string] [-run_from_directory]
     [-unix_shell] [design]
```

Writes out data and script files for Modus to perform Logic Built-in Self Test.

This command generates the following files for an LBIST macro that is JTAG-controlled:

- *topmoduleName*.et_netlist.v—A netlist for Modus
- assignfile.JTAG.*instructionName*—A pin-assignment file for the parent mode for RUNBIST/SETBIST instruction
- assignfile.LBIST.*instructionName*—A pin-assignment file for the child mode for RUNBIST/SETBIST instruction
- MODE_JTAG_*instructionName*—A mode definition file that describes the testmode in parent mode for RUNBIST/SETBIST instruction
- MODE_LBIST—A mode definition file that describes the testmode in child mode for LBIST
- TBDseqpatt.JTAG_*instructionName*—A sequence definition file for parent mode for RUNBIST/SETBIST instruction
- TBDseqpatt.LBIST_*instructionName*—A sequence definition file for child mode for RUNBIST/SETBIST instruction
- TestSequence.seq—Test sequence file for LBIST test
- run_lbist_*instructionName*—A Modus file to run LBIST tests for RUNBIST/SETBIST instruction
- run_sim_*instructionName*—An Xcelium simulator run file used to simulate the test patterns created by Modus for LBIST.

This command generates the following files for an LBIST macro using the (non-programmable) direct access interface:

- *topmoduleName*.et_netlist.v—A netlist for Modus
- assignfile.MODE_LBIST_DIRECT—A pin-assignment file for the LBIST testmode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `assignfile.NCSIM`—A pin assignment file for the testmode used to generate Verilog to verify the LBIST signature
- `MODE_LBIST`—A mode definition file that describes the testmode for LBIST
- `TBDseqpatt.MODE_LBIST_DIRECT`—A sequence definition file for LBIST mode
- `TBDpatt.NCSIM`— A sequence definition file for the testmode used to generate Verilog testbench to verify the LBIST signature
- `TestSequence.seq`—Test sequence file for LBIST tests
- `run_lbist_DIRECT`—A Modus file to run direct-access LBIST tests
- `run_sim_DIRECT`—An Xcelium simulator run file used to simulate the test patterns created by Modus for direct-access LBIST.
- `run_ET_NCSIM`—A Modus file to generate the Verilog testbench to verify the LBIST signature

This command generates the following files for an LBIST macro using the programmable direct access interface:

- *`topmoduleName`*`.et_netlist.v`—A netlist for Modus
- `assignfile.BASE_PGM_DIRECT`—A pin-assignment file for the parent mode for programmable direct-access LBIST
- `assignfile.LBIST_PGM_DIRECT`—A pin assignment file for the child mode for programmable direct-access LBIST
- `MODE_BASE_PGM_DIRECT`—A mode definition file that describes the testmode in parent mode for programmable direct-access LBIST
- `MODE_LBIST`—A mode definition file that describes the testmode for LBIST
- `TBDseqpatt.BASE_PGM_DIRECT`—A sequence definition file for parent mode for programmable direct-access LBIST
- `TBDseqpatt.LBIST_PGM_DIRECT`—A sequence definition file for child mode for programmable direct-access LBIST
- `TestSequence.seq`—Test sequence file for LBIST tests
- `run_lbist_PGM_DIRECT`—A Modus file to run direct-access LBIST tests for programmable direct-access LBIST
- `run_sim_PGM_DIRECT`—An Xcelium simulator run file used to simulate the test patterns created by Modus for programmable direct-access LBIST.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- `run_ET_NCSIM`—A Modus file to generate the Verilog testbench to verify the LBIST signature

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-build_faultmodel_options {-option1 value -option2 value...}`

Specifies a string containing the extra options to build a fault model.

Note: For more information on these options, refer to the `build_faultmodel` command in the *Command Line Reference* (of the Modus documentation).

`-build_model_options {-option1 value -option2 value...}`

Specifies extra options to apply when building the model.

Note: For more information on these options, refer to the `build_model` command in the *Command Line Reference* (of the Modus documentation).

`-build_testmode_options {-option1 value -option2 value...}`

Specifies extra options to apply when building the test mode.

Note: For more information on these options, refer to the `build_testmode` command in the *Command Line Reference* (of the Modus documentation).

design

Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *string*

Specifies the directory to which the output files must be written.

Default: *current_working_directory*/`lbist_testbench_scripts`

`-library` *string*

Specifies the list of Verilog structural library files required to run Modus.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can specify the files explicitly or you can specify an *include* file that lists the files. You can also specify directories of Verilog files but you cannot reference directories in an include file.

For example, if the following Verilog files are required:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_et_lbist` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_et_lbist -library "./padcells.v ./stdcells.v \
 ./memories ./ip_blocks" ...
```

2. If using an include file. Create an include file named `include_libraries.v` containing:

```
`include "./padcells.v"
`include "./stdcells.v"
```

And then specify the following:

```
write_et_lbist -library "include_libraries.v ./
memories \
 ./ip_blocks" ...
```

Note: If you specify a relative path, the command interprets the path to be the location from where Modus will be run.

`-ncsim_library` *`string`*

Specifies the list of library files required for the Xcelium simulation.

`-run_from_directory`

Allows to perform Logic Built-in Self Test from the working directory specified with the `-directory` option.

Default: parent directory of the specified work directory.

`-unix_shell` Generates the files for use with Encounter Test.

`-verify_test_structures_options {-`*`option1 value -option2 value`*`}`

Specifies extra options to apply when performing test structure verification.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

See the following sections in *Genus Design for Test Guide for Legacy UI*:

- Inserting LBIST Logic
- Generating Files for LBIST Pattern Generation and Simulation

Affected by these commands: insert_dft logic_bist on page 1051
write_logic_bist_macro on page 1201

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_et_no_tp_file

```
write_et_no_tp_file
     [-min_slack integer] [design]
```

Generates a d*esign*.noTpFile file which contains a list of subdesigns, instances, nets or pins that have been constrained. As a result, no test points can be inserted on these objects.

This file is passed to Modus via the -notpfile option of the analyze_random_resistance and analyze_deterministic_faults commands. With this information, Modus does not generate test points for these subdesigns, instances, pins or nets, thereby providing a better set of test points.

Subdesigns and instances included in this file satisfy any of the following conditions:

- The subdesign or instance has the dft_dont_scan attribute set to true
- The subdesign or instance has the preserve attribute set to true, 1, or size_ok
- The instance has the hard_region attribute set to true
- The subdesign or instance has been scan abstracted (using either native or CTL model)

The pins that are included in the file satisfy any of the following conditions:

- The pin has a slack less than the min_slack
- The pin has the preserve attribute set to true, 1, or size_ok

In addition, any net that does not have its preserve attribute set to false, is written to the file.

If a subdesign is already written into the file, then the instances, pins or nets in the subdesign are not written out again. Similarly, if the instance is included in the file, then its pins and nets are not written out to the file.

Options and Arguments

design	Specifies the design on which to perform test point selection. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-min_slack` *`integer`*

Prevents insertion of a test point if the slack is smaller than the specified number (in ps).

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_io_speclist

```
write_io_speclist > iospeclist_file
     [-supplemental_file file]
```

Writes out the IOSpecList output file. The IOSpeclist output file describes the boundary scan architecture of the design. The file contains all ports (functional, test, and TAP) in the design, specifies the type and location of the boundary cells to be inserted on all the functional ports, and lists the instructions (both mandatory and user-defined) to be built in the JTAG Macro.

Options and Arguments

iospeclist_file Specifies the name of the file to be written.

-supplemental_file *file*

Specifies the name of the supplemental file to write out. This file and its corresponding IOSpeclist file are used to define the boundary scan objects prior to inserting boundary scan logic. The supplemental file lists the boundary scan segments and the JTAG-instruction definitions for its related objects written to the IOSpeclist file. When using an IOSpeclist file to define the order of the boundary scan register, the supplemental file should be included into the Genus session before reading the IOSpeclist input file. Both the supplemental and the IOSpeclist files need only be written if your intention is to insert boundary-scan logic in a new Genus session.

Note: The recommended approach to completely restoring the DFT setup (including the boundary scan objects) in a new Genus session is to use the `write_script`/`read_netlist` approach.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Writing the IOSPecList in *Genus Design for Test Guide for Legacy UI*

Affected by these commands:	define_dft jtag_instruction on page 933
	define_dft jtag_instruction_register on page 937
	insert_dft boundary_scan on page 1026
	insert_dft ptam on page 1071
Related command:	read_io_speclist on page 1116

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_logic_bist_macro

```
write_logic_bist_macro
     -channels integer -max_length_of_channels integer
     [-clocks integer]
     [-add_runbist_support] [-add_setbist_support]
     [-add_direct_access_support]
     [-add_programmable_direct_access_support]
     [-add_masking]
     [-scan_patterns integer] [-scan_pattern_counter_length integer]
     [-set_patterns integer] [-set_pattern_counter_length integer]
     [-reset_patterns integer] [-reset_pattern_counter_length integer]
     [-static_patterns integer] [-static_pattern_counter_length integer]
     [-dynamic_patterns integer]  [-dynamic_pattern_counter_length integer]
     [-scan_channels_counter_length integer]
     [-scan_window_ integer] [-scan_window_counter_length integer]
     [-scan_window_pulse_value integer]
     [-scan_enable_delay integer] [-scan_enable_delay_counter_length integer]
     [-capture_window integer] [-capture_window_counter_length integer]
     [-capture_window_capture_value integer]
     [-capture_window_launch_value integer]
     [-set_reset_test_window integer] [-set_reset_pulse_width integer]
     [-capture_clock_stagger_ratio integer]
     [-scan_clock_stagger_ratio integer]
     [-parallel_load_control_bits] [-parallel_load_prpg]
     [-parallel_load_misr] [-parallel_load_mask]
     [-info_file string] [> file]
```

Writes out the structural netlist using generic logic for the LBIST macro. The generated macro has a PRPG, MISR, 1149.1 Interface, and a BIST Controller finite state machine. If you do not insert an LBIST macro with direct access support, the macro can be initialized by the JTAG macro using the `RUNBIST` and `SETBIST` instructions.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

For more information about LBIST, contact your local Cadence representative.

Options and Arguments

`-add_direct_access_support`

Inserts the LBIST macro with (non-programmable) direct-access support.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-add_masking`
: Inserts the LBIST macro including the optional channel masking logic (only available with SETBIST support or programmable direct-access support.).

  Note: This option can only be specified with the `-add_setbist_support` or `-add_programmable_direct_access_support` option.

`-add_programmable_direct_access_support`
: Inserts the LBIST macro with programmable direct-access support.

`-add_runbist_support`
: Inserts the LBIST macro with JTAG RUNBIST support.

`-add_setbist_support`
: Inserts the LBIST macro with JTAG SETBIST support.

`-capture_clock_stagger_ratio` *`integer`*
: Controls staggering of the capture clocks. The specified value determines the number of bistck pulses after which the capture clock is staggered. Specify 0, 1, 2, 4 or 8. If you specify 0, the capture clocks will not be staggered.

`-capture_window` *`integer`*
: Specifies the default value of the capture window.

`-capture_window_capture_value` *`integer`*
: Specifies the value of the capture window (down) counter at which the (first) capture clock pulse is issued.

`-capture_window_counter_length` *`integer`*
: Specifies the length of the capture window counter that is part of the LBIST macro.

`-capture_window_launch_value` *`integer`*
: Specifies the value of the capture window (down) counter at which the launch clock pulse is issued.

`-channels` *`integer`*
: Specifies the number of compressed scan channels. It can include additional channels required to include the boundary scan chain in the macro too.

`-clocks` *`integer`*
: Specifies the number of staggered scan and capture clocks.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dynamic_patterns` *`integer`*

Specifies the default number of dynamic test patterns to be executed.

`-dynamic_pattern_counter_length` *`integer`*

Specifies the length of the dynamic pattern counter that is part of the LBIST macro.

`-info_file` *`file`*

Specifies the file that contains more detailed information about the LBIST macro. It has details of the MISR and PRPG that can be used when writing out the scripts to run Modus.

`-max_length_of_channels` *`integer`*

Specifies the length of the longest compressed scan channel.

`-parallel_load_control_bits`

Makes the LBIST control bits programmable at run-time through the programmable direct-access LBIST parallel load interface. For more information on the LBIST control bits refer to the LBIST chapter in *Genus Design for Test Guide for Legacy UI*.

`-parallel_load_mask`

Makes the LBIST mask programmable at run-time through the programmable direct-access LBIST parallel load interface.

Note: This option can only be specified with the `-add_masking` option.

`-parallel_load_misr`

Makes the LBIST MISR programmable at run-time through the programmable direct-access LBIST parallel load interface.

`-parallel_load_prpg`

Makes the LBIST PRPG programmable at run-time through the programmable direct-access LBIST parallel load interface.

`-reset_patterns` *`integer`*

Specifies the default number of reset test patterns to be executed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-reset_pattern_counter_length` *`integer`*

Specifies the length of the reset pattern counter that is part of the LBIST macro.

`-scan_channel_counter_length` *`integer`*

Specifies the length of the scan channel counter that is part of the LBIST macro.

`-scan_clock_stagger_ratio` *`integer`*

Controls staggering of the scan clocks. The specified value determines the number of bistck pulses after which the scan clock is staggered. Specify 0, 1, 2, 4 or 8. If you specify 0, the scan clocks will not be staggered.

`-scan_enable_delay` *`integer`*

Specifies the default delay after the scan enable signal toggles.

`-scan_enable_delay_counter_length` *`integer`*

Specifies the length of the scan enable delay counter that is part of the LBIST macro.

`-scan_patterns` *`integer`*

Specifies the default number of scan test patterns to be executed.

`-scan_pattern_counter_length` *`integer`*

Specifies the length of the scan pattern counter that is part of the LBIST macro.

`-scan_window` *`integer`*

Specifies the default value of the scan window.

`-scan_window_counter_length` *`integer`*

Specifies the length of the scan window counter that is part of the LBIST macro.

Default: `8`

`-scan_window_pulse_value` *`integer`*

Specifies the value of the scan window (down) counter at which the (first) scan clock pulse is issued.

`-set_patterns` *`integer`*

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specifies the default number of set test patterns to be executed.

`-set_pattern_counter_length` *`integer`*

Specifies the length of the set pattern counter that is part of the LBIST macro.

`-set_reset_pulse_width` *`integer`*

Specifies the width of the asynchronous set/reset pulse for the set/reset tests.

`-set_reset_test_window` *`integer`*

Specifies the value of the test window for the asynchronous set/reset tests.

`-static_patterns integer`

Specifies the default number of static test patterns to be executed.

`-static_pattern_counter_length integer`

Specifies the length of the static pattern counter that is part of the LBIST macro.

Related Information

Inserting LBIST Logic in *Genus Design for Test Guide for Legacy UI*

Related commands: insert_dft logic_bist on page 1051

write_dft_lbist_test_points on page 1173

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_memory_physical_map

```
write_memory_physical_map
     -memory string [-directory string] [design]
```

Writes the physical memory layout of the memory cell provided with the -memory option to a file named *ex_memory*.mca in a directory provided by the -directory option.

The physical memory layout can be generated only after reading memory information via read_memory_view command.

This *ex_memory*.mca file represents the physical layout of the memory, that is, how the memory is physically structured into banks, data-bits, columns and rows. It also depicts the order of banks, order of rows and columns in a bank etc. The format of this layout structure matches the format given in the documentation under address_partition section in *Genus PMBIST Guide for Legacy UI*.

Options and Arguments

design	Specifies the design on which to perform the memory layout. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
-memory *string*	Specifies the name of the memory module in the specified design.
-directory *string*	Specifies the directory path where to store the memory file created. Default: ./pmbist_*designname*

Related Information

Affected by this command: read_memory_view on page 1117

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_pmbist_interface_files

```
write_pmbist_interface_files
     -directory string [-rtl] [design]
```

Writes out the necessary interface files for programmable Memory Built-In-Self-Test (PMBIST) related to this design.

Genus optimizations and changes to the design which affect the data within the internal representation of the interface files update these files until written. Therefore, you should execute the command just prior to writing out a design and starting PMBIST design verification.

These interface files represent an abstract model of the current PMBIST insertion process, supporting not only a bottom-up flow for incremental PMBIST insertion but also the generation of patterns to exercise the PMBIST logic from Modus `create_embedded_test` command.

This command generates the following files:

- *design*`_pattern_control.txt`—A file containing the PMBIST external interface and pattern generation controls derived from the configuration file.
- *design*`_test_def.txt`—A file containing information on testplans, algorithm constraints, and user-defined algorithms for this design.
- *design*`_mbistsch_tdr_map.txt`—A file identifying the `MBISTSCH TDR` test data register contents.
- *design*`_mbistchk_tdr_map.txt`—A file identifying the `MBISTCHK TDR` test data register contents.
- *design*`_mbisttpn_tdr_map.txt`—A file identifying the `MBISTTPN TDR` test data register contents.
- *design*`_mbistamr_tdr_map.txt`—A file identifying the `MBISTAMR TDR` test data register contents when programmable testplans ate required.
- *design*`_mbistrar_tdr_map.txt`—A file identifying the `MBISTRAR TDR` test data register contents. This is generated when redundancy analysis or fault tolerance is requested. This is used to determine the failure details for redundancy analysis, fault tolerance, and repair.
- *design*`_mbistdiag_tdr_map.txt`—A file identifying the `MBISTDIAG TDR` test data register contents. This is generated when diagnostics is requested for memories.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- *design*_mbistrom_tdr_map.txt—A file identifying the MBISTROM TDR test data register contents when ROMs are present in the design and programmable testplans target them.
- *design*_mbistromdiag_tdr_map.txt—A file identifying the MBISTROMDIAG TDR test data register contents when ROM diagnostics is required.

If hard repair is also performed then this command generates the following additional files:

- *design*_fcu_pattern_control.txt—A file containing the PMBIST hard repair specific external interface and pattern generation controls derived from the configuration file.
- *design*_fcu_test_def.txt—A file containing information on testplans, algorithm constraints, and user-defined algorithms for this design for hard repair.
- *design*_fcusch_tdr_map.txt—A file identifying the FCUSCH TDR test data register contents. This is used to schedule the fuse macro for hard repair.
- *design*_fcuchk_tdr_map.txt—A file identifying the FCUCHK TDR test data register contents. This is used to check the results of the hard repair.
- *design*_fcutpn_tdr_map.txt—A file identifying the FCUTPN TDR test data register contents. This contains information on testplans used for hard repair.
- *design*_fcuamr_tdr_map.txt—A file identifying the FCUAMR TDR test data register contents when programmable testplans are required for hard repair.
- *design*_fcurr_tdr_map.txt—A file identifying the FCURR TDR test data register contents. This contains information on channel binding for hard repair.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

design	Specifies the design for which to write out the PMBIST interface files. If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
-directory *string*	Specifies the directory to which the PMBIST interface files must be written.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-rtl`	Writes out the PMBIST interface files for the RTL flow. The interface files for RTL flow may be different than the gate level flow due to addition of labels for `for` and `if`.

Related Information

Affected by these commands:	define_dft pmbist_direct_access on page 958
Affects this command:	write_pmbist_interface_files on page 1207
Related command:	read_pmbist_interface_files on page 1120

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_pmbist_testbench

```
write_pmbist_testbench
     [-create_embedded_test_options string]
     [-create_fcu_patterns] [-add_asu_testbench]
     [-irun_options string] [-sim_options string] [-ncsim_library file_list]
     [-testbench_directory string] [-directory string]
     [-no_deposit_script | -deposit_all_flops]
     [-amu_instance hinst] [-fcu_instance hinst]
     [-sim_with_deposit_script] [-aux_source string] [-aux_top string]
     [no_build_model] [-script_only][-create_fcu_patterns]
     [-test_pattern_directory string][-rtl] [design]
```

Writes out the necessary files and the template run scripts to create Verilog test benches to validate PMBIST, and executes the scripts using the Modus software.

If your design contains ROMs, the `rompath` and `romcontentsfile` keywords in the `create_embedded_test_options` string are set by default.

Typically a single memory BIST instruction set is implemented in the design. In this case, the `interfacefilelist` keyword to `create_embedded_test` is set by default. If multiple memory BIST instruction sets are implemented in the design, include the `interfacefilelist` keyword indicating the interface files for a single instruction set in the `create_embedded_test_options` string.

After inserting PMBIST into the design and optionally a JTAG macro, write out the modified design to file. Then, use this command to generate a PMBIST Verilog test bench to validate the PMBIST functionality through simulation.

This command is designed to generate the following patterns by default:

- `production` patterns— if JTAG control has been implemented for memory BIST in the design
- `directaccess` and `burnin` patterns—if direct access has been implemented for PMBIST in the design

These default generated test benches are created with a schedule intended to run all devices in parallel.

This command generates the following files prior to executing the run scripts in Modus:

- *`design`*`_abstract.v`—A Verilog generic logic gates description of the netlist for Modus
- `runmodus.write_mbist_testbench.tcl`—A template script file to run `create_embedded_test` in Modus, generating patterns for PMBIST. This is

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

automatically executed by `write_pmbist_testbench`, unless the `-script_only` option has been specified.

- `runmodus.write_vectors.tcl`—A template script file to run `write_vectors` in Modus to write the PMBIST patterns as Verilog test benches. This is automatically executed by `write_pmbist_testbench`, unless the `-script_only` option has been specified. If the customexperiment option is specified in the `-create_embedded_test_options` keyword, then this script will be suffixed with the experiment name. Example, if `-create_embedded_test_options "customexperiment=myexp"` is specified, then the script name will be `runmodus.write_vectors.tcl.myexp`.

- `irun.simscript.`*`testbench_pattern_name`*—`irun.simscript.`*`testbench_pattern_name`*—One or more template scripts to compile and simulate the PMBIST test bench pattern within the Xcelium simulator. By default, these scripts will be for production patterns when JTAG is used and for `directaccess` and `burnin` patterns when direct access is used. These scripts are created in the directory specified by the `-testbench_directory` keyword. If a directory is not specified, it defaults to an `mbist_testbench` directory created in the directory specified by the `-directory` option.

- `irun.depositscript.`*`testbench_pattern_name`*—One or more Xcelium simulator scripts to initialize all PMBIST and JTAG logic to known states prior to executing the Verilog test benches. Genus logic optimizations may permit "X" state propagation. These scripts can be used to overcome this situation when desired by including `-input` keyword selecting this file in the appropriate irun script. Note these scripts include simulator run and exit commands within them. This file is generated by default, use the `-no_deposit_script` option to not generate this file. By default the `irun.simscript.`*`testbench_pattern_name`* scripts will not simulate using the `irun.depositscript.`*`testbench_pattern_name`* file. Specify the `-sim_with_deposit_script` to simulate with the deposit script.

You can customize some files according to the setup requirements.

Note: To use this command you need an additional license. For more information on the exact product requirements, refer to Encounter Test or Modus Product Requirements for Advanced Features in *Genus Design for Test Guide for Legacy UI*.

Options and Arguments

`-add_asu_testbench`	Specifies to enable generation of the ASU Verilog driver for testing on demand functions.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-amu_instance` *`hinst`*

Specifies the name of the associated algorithm memory unit instance.

`-aux_source` *`string`* Specifies a list of auxiliary design files for simulation. For example, `"mod1.v mod2.v"`

`-aux_top` *`string`* Specifies a list of auxiliary top level design modules for simulation. For example, `"mod1 mod2"`

`-create_embedded_test_options {`*`-option1 value -option2 value`*`}`

Specifies extra options to apply when running Create Embedded Test.

The default options are to generate bypass and production patterns (-createpatterns bypass,production) if MBIST is controlled by JTAG and directaccess/burnin patterns (-createpatterns directaccess, burnin) if MBIST is controlled by direct access pins in parallel schedules.

`-create_fcu_patterns`

Creates patterns for hard repair. By default, `write_pmbist_testbench` creates MBIST patterns for logic inserted by `insert_dft pmbist`. If hard repair support has been added by running `insert_dft hard_repair`, then use this option to generate hard repair patterns. Hard repair patterns and non hard repair patterns can't be generated at the same time.

When -`create_fcu_patterns` is used, the `customexperiment` and `customtestplans` options must be specified in the -`create_embedded_test_options` keyword. Also, only one hard repair pattern type can be created at a time.The `createpatterns` keyword must also be specified, and set to either `jtag_fcu_repair` (for JTAG) or `pmda_fcu_repair` (for direct access).

Example:
```
write_pmbist_testbench
-create_fcu_patterns
-create_embedded_test_options
"customexperiment=myexp customtestplans=
tp1,tp2,tp3 createpatterns=jtag_fcu_repair"
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-deposit_all_flops`
: Deposits a value for all flops in the deposit script. This can be useful when it's necessary to ensure a known value (non X value) is present in all sequential elements prior to starting simulation.

`design`
: Specifies the design for which to write out the Modus input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-directory` *`string`*
: Specifies the directory to which the output files (scripts, test data and abstract model) must be written.

Default: *`current_working_directory`*`/wmt`

`-fcu_instance` *`hinst`*
: Specifies the name of the associated fuse control unit instance.

`-irun_options {`*`-option1 value -option2 value...`*`}`
: Specifies the options to be passed to the `irun` command for simulation purposes. These options will be placed at the beginning of the irun script, and indicated by the comments "## User specified options:".

`-ncsim_library` *`file_list`*
: Specifies the list of library files required for the Xcelium simulation.

You can specify the files explicitly or you can specify an include file that lists the files. You can also specify directories of Verilog files but you cannot reference directories in an include file.

For example, if the Verilog files required to run simulation are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_pmbist_testbench` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_pmbist_testbench -ncsim_library "./padcells.v
./stdcells.v ./memories ./ip_blocks" ...
```

2. If using an include file. Create an include file named `include_libraries.v` containing:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
`include "./padcells.v"
`include "./stdcells.v"
```

And then specify the following:

```
write_pmbist_testbench -ncsim_library \
"include_libraries.v ./memories  ./ip_blocks" ...
```

Note: If you specify a relative path, the command interprets the path to be the location from where the Xcelium simulator will be run.

`-no_build_model`

Prevents to execute the `build_model` command.

This can be useful if the Modus model has already been built by a prior `write_pmbist_testbench` invocation and does not need to be built again, especially if it's a large design and takes a significant amount of time to build.

`-no_deposit_script`

Prevents that the deposit script is generated and used during simulation.

`-rtl`

Targets the necessary RTL files for simulation in the irun script for the RTL flow.

`-script_only`

Creates the `runmodus.write_mbist_testbench.tcl` script without executing it.

By default, the tool creates and executes the `runet.write_mbist_testbench` script, and then creates and executes the `runet.write_vectors` script.

`-sim_options {-option1 value1 -option2 value2 ...}`

Specifies the options to be passed to the simulation script.

`-sim_with_deposit_script`

Specifies to run simulation with a deposit script. Genus logic optimizations may permit "X" state propagation. Simulating with a deposit script can be used to overcome this situation when desired.

`-testbench_directory directory`

Specifies the directory to which the generated Verilog test benches must be written.

Default: `current_working_directory/mbist_testbench`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-test_pattern_directory` *`string`*

Specifies the output directory for TBDpatt files, assign files and mode initialization files.

Examples

- In the following example a single PMBIST instruction set was implemented in the design:

```
write_pmbist_testbench -testbench_directory ./mbist_verilog_testbenches \
    -ncsim_library ../$simulation_verilog_libraries \
    -directory ./mbist_verification_scripts
```

- The following example assumes you are working in a multiple block flow (PMBIST was inserted on multiple designs or blocks) using separate JTAG instructions to access each block's MBIST engines. When merged into the TOP level, each instruction set is labeled with a unique integer suffix by default.

```
write_pmbist_testbench \
    -create_embedded_test_options \
    interfacefilelist="TOP_1_pattern_control.txt, \
    TOP_1_mbisttpn_tdr_map.txt,TOP_1_mbistsch_tdr_map.txt \
    TOP_1_mbistchk_tdr_map.txt,TOP_1_test_def.txt" \
    -testbench_directory ./mbist_verilog_testbenches \
    -ncsim_library../simulation_verilog_libraries     \
    -directory ./mbist_verification_scripts
```

Related Information

Affected by these commands:	define_dft pmbist_direct_access on page 958
	insert_dft boundary_scan on page 1026
	insert_dft jtag_macro on page 1046
	insert_dft pmbist on page 1065
Related commands:	read_hdl on page 336
	write_pmbist_interface_files on page 1207
Related attribute:	pmbist_instruction_set

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_scandef

```
write_scandef
     [-partition partition -chains chain [chain]...]...
     [-version {5.5|5.4}]
     [-end_chains_before_lockups]
     [-dft_configuration_mode dft_config_mode_name]
     [-dont_split_by_library_domains]
     [-dont_split_by_power_domains]
     [-dont_use_timing_model_pins] [design] [ > file]
```

Writes the scanDEF description of the top-level scan chains configured in the design for reordering using a physical design tool.

Options and Arguments

`-chains` *chain*

Specifies the scan chains that must be grouped in the same partition by the physical design tool. Use the `report_dft_chains` command to obtain a list of valid chain names.

The tool ensures that chains or chain segments that are not compatible are not added to the same partition, but are further partitioned by adding the test clock name and test clock edge to the final partition name.

Note: Requires version 5.5.

design

Specifies the design for which to write out the scanDEF description.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_configuration_mode` *dft_configuration_mode_name*

Specifies the configuration mode for which to write the scan definition

`-dont_split_by_library_domains`

Indicates not to split the chains at the scan data input pin of the last flop in the originating (or from) library domain and at the scan data output pin of the first flop in the destination (or to) library domain.

By default, the chains will be split based on the library domains. If the design has no library domains, the chains will not be split.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dont_split_by_power_domains`

Indicates not to split the chains at the scan data input pin of the last flop in the originating (or from) power domain and at the scan data output pin of the first flop in the destination (or to) power domain.

By default, the chains will be split based on the power domains. If the design has no power domains, the chains will not be split.

`-dont_use_timing_model_pins`

Prevents using the user-designated libcell timing model pins as the scanDEF chain `START` and `STOP` points. Instead an outward trace is performed to identify and use the first flip-flop scan data output pin and last flip-flop scan data input pin and use these pins as `START` and `STOP` pins in the scanDEF chains.

`-end_chains_before_lockups`

Terminates the scan segment at the scan data input pin of the scan flop which precedes the lockup element in the scan DEF chain.

`file`

Specifies the file to which the output must be written. To write out the scanDEF file in compressed format, specify a file name with the `.gz` extension.

Default: output is written to the screen

`-partition` *`partition`*

Specifies the name of a user-defined partition.

Note: Requires version 5.5.

`-version {5.5|5.4}`

Specifies which DEF version to write out. Version 5.5 writes out the `MAXBITS` and `PARTITION` keywords as regular statements (that is, uncommented).

Default: `5.5`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example writes out the scanDEF information to the screen:

```
legacy_genus:/> write_scandef

VERSION 5.4 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN top ;

SCANCHAINS 2 ;
 - chain_1
 + START u_a/out_reg_1 Q
 + FLOATING
  u_a/out_reg_2 ( IN SI ) ( OUT Q )
  u_a/out_reg_3 ( IN SI ) ( OUT Q )
 + STOP buf_2 A
 ;

 - chain_2
 + START buf_1 Y
 + FLOATING
  u_b/out_reg_0 ( IN SI ) ( OUT Q )
  u_b/out_reg_1 ( IN SI ) ( OUT Q )
 + STOP u_b/out_reg_3 SI
 ;

END SCANCHAINS
END DESIGN
```

Related Information

Affected by this command:	connect_scan_chains on page 906
Affected by these attributes:	Actual Scan Chain attributes

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

14

Low Power Synthesis

- clock_gating on page 1221
- clock_gating add_xor_gating on page 1223
- clock_gating connect_test on page 1225
- clock_gating declone on page 1226
- clock_gating import on page 1228
- clock_gating insert_in_netlist on page 1230
- clock_gating insert_obs on page 1231
- clock_gating join on page 1233
- clock_gating rebalance on page 1235
- clock_gating remove on page 1236
- clock_gating share on page 1238
- clock_gating split on page 1240
- read_saif on page 1242
- read_tcf on page 1247
- read_vcd on page 1252
- report clock_gating on page 1256
- report power on page 1257
- state_retention on page 1258
- state_retention connect_power_gating_pins on page 1259
- state_retention swap on page 1260
- write_forward_saif on page 1261

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- write_saif on page 1263
- write_tcf on page 1265

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating

```
clock_gating
     { add_xor_gating | connect_test | declone | import | insert_in_netlist
     | insert_obs | join | remove | share | split}
```

Manipulates a netlist for clock gating.

Important

The `clock_gating` commands only work on a mapped netlist.

Options and Arguments

`add_xor_gating`	Performs XOR clock-gating.
`connect_test`	Connects the test input of all clock-gating logic.
`declone`	Merges clock-gating instances driven by the same inputs.
`import`	Processes clock-gating instances that were either manually inserted or inserted by third-party tools to make them recognizable as clock-gating instances by the Genus-LP engine.
`insert_in_netlist`	Inserts clock-gating logic on a mapped netlist.
`insert_obs`	Inserts and connects observability logic.
`join`	Joins multiple-stage clock-gating logic into a single clock-gating instance with a complex enable function.
`rebalance`	Regroups the flops of the clock-gating instances driven by the same inputs based on the values of the `lp_clock_gating_min_flop` and `lp_clock_gating_max_flops` attributes.
`remove`	Removes the specified clock-gating logic.
`share`	Extracts the enable function shared by clock-gating logic and inserts shared clock-gating logic with the common enable sub function as the enable signal.
`split`	Splits a single clock-gating instance with a complex enable function into multiple stages of clock-gating logic.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related commands:

clock_gating add_xor_gating on page 1223
clock_gating connect_test on page 1225
clock_gating declone on page 1226
clock_gating import on page 1228
clock_gating insert_in_netlist on page 1230
clock_gating insert_obs on page 1231
clock_gating join on page 1233
clock_gating rebalance on page 1235
clock_gating remove on page 1236
clock_gating share on page 1238
clock_gating split on page 1240

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating add_xor_gating

```
clock_gating add_xor_gating
     -flops flops
     [-skip_async_check]
     [-max_distance integer] [-threshold integer]
     [-do_placement]
     [-detailed_log file]
```

Takes a list of flops and performs XOR clock-gating. You can use this command in the logical or physical flow. It is recommended to call this command after mapping but before DFT scan chain connection and before multi-bit inferencing.

This command groups the flops according to

- the clock source
- the flop type
- the minimum/maximum constraints
- the module
- the physical distance

Following flops will not be considered for XOR clock-gating:

- Complex flops—Flop has multiple D, Clk pins or flop which is hard to analyze
- Scan flops used as functional flops
- Multi-bit flops
- If nets around the flop's D, Q and clock pin is preserved
- Flop is already Genus clock-gated
- Flop is timing critical—After adjusting the estimated delay of XOR structure, the slack of the D pin of the flop should be non-negative.
- Synchronous control pins are connected to non-constant value
- Asynchronous control pin of the flop must be connected to non-constant value. This check can be skipped using the `-skip_async_check` option.
- Flop is marked excluded from clock gating with the `lp_clock_gating_exclude` attribute

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Options and Arguments

`-detailed_log` *`file`*	Specifies the file name to which the detailed log information must be written.
`-do_placement`	Places the xor flops in the physical flow. **Note:** This option will work only when the design is fully placed. For partially-placed design, tool will ignore this option.
`-flops` *`flops`*	Identifies the list of flops that require xor clock-gating.
`-max_distance` *`integer`*	Groups the flops in clusters depending upon the physical location. Each cluster has at least one pair of flops whose distance will be less than the specified distance. **Note:** This option is ignored in non-physical flows. *Default*: The tool calculates the distance depending on the library loaded.
`-skip_async_check`	Specifies not to perform the asynchronous control signal check for the list of flops specified. **Note:** XOR clock-gating will not be performed on the flops which do not have asynchronous control signal; mention this option if you need to perform XOR clock-gating on such flops.
`-threshold` *`integer`*	Specifies the minimum number of flip-flops to be gated.

Related Information

Related command:	syn_map on page 529
	lp_clock_gating_exclude
Related attributes:	lp_clock_gating_max_flops
	lp_clock_gating_min_flops

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating connect_test

```
clock_gating connect_test
```

Globally connects the test input of all clock-gating logic to the test signal specified through the `lp_clock_gating_test_signal` attribute and marks this network as ideal. This command applies to the current design or the current hierarchical instance.

If the clock-gating test input is already connected, the command has no effect.

Note: This command works only on a netlist whose clock-gating logic was inserted by the Genus-LP engine. If clock-gating was inserted by a third party tool, the command clock_gating import needs to be run prior to this command.

Note:

Example

- The following command connects the test inputs connected to Test1.

```
syn_generic
...
set_attr lp_clock_gating_test_signal Test1
...
clock_gating connect_test
```

Related Information

Related command:	report clock_gating on page 1256
Affected by the attribute:	lp_clock_gating_test_signal

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating declone

```
clock_gating declone
     [-hierarchical] [-no_clock_tree_traversal]
     [-verbose]
```

If the clock-gating logic of different registers in the design uses the same enable signal, Genus can merge these clock-gating instances for any such identically gated registers

The Genus-LP engine automatically removes any dangling ports.

Note: This command works only on a netlist whose clock-gating logic was inserted by the Genus-LP engine. It will not work on the clock gating logic created by other third party tools. After decloning, it will create the new instance named RC_CG_DECLONE_HIER_INST out of RC_CG_HIER_INSTs in the netlist.

The logfile also shows the information such as:

```
Info : Clock-gating instances are decloned. [POPT-56]
     : New decloned instance is gdfcc_0/gdhb_data_out_ctrl_0/
     RC_CG_DECLONE_HIER_INST
```

Old instances were:

```
     gdfcc__0/gdfcc_b_data_out_ctrl_0/RC_CG_HIER_INST19
     gdfcc__0/gdfcc_b_data_out_ctrl_0/RC_CG_HIER_INST24

     Clock-gating declone status
     ===========================
     Total number of clock-gating instances before: 128
     Total number of clock-gating instances after : 64
```

Check for the [POPT-56] messages in the logfile to see if the decloning has happened or not.

Options and Arguments

`-hierarchical`
: Allows traversing the design hierarchy to search for clock-gating instances that can be merged. As a result, new ports can be added for the gated-clock signal.

  By default, this command only affects instances at the current level of the hierarchy.

`-no_clock_tree_traversal`
: Prevents traversing through buffers and inverter pairs on the clock signal.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	If you do not set this option, Genus can remove buffers or inverter pairs on the clock path during optimization unless the buffers or inverter pairs are marked preserved.
`-verbose`	Includes detailed information.

Related Information

Related command:	report clock_gating on page 1256
Affected by this attribute:	lp_clock_gating_max_flops

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating import

```
clock_gating import
     [-start_from instance] [-hierarchical]
     [-keep_name] [-detail [-verbose]]
```

Processes clock-gating instances that were either manually inserted or inserted by third-party tools to make them recognizable as clock-gating instances by the Genus-LP engine.

The command returns the total number of instances imported.

Currently, the Genus-LP engine recognizes the following structures as clock-gating instance:

- Two-input AND or NAND gates that have a clock signal driving one of the inputs

  In this case, the other pin is assumed to be the enable pin.

  Note: If the other pin is part of the test network, the Genus-LP engine does not recognize the gate as a clock-gating instance.

- Integrated clock-gating cells

Currently, the Genus-LP engine does not recognize these structures as clock-gating instances if they were defined in a separate module that is instantiated in the netlist.

The Genus-LP engine creates a new hierarchical instance (RC_CG_HIER_INST) for each clock-gating instance it recognizes and adds it to the list of clock-gating instances that the Genus-LP engine has created.

Options and Arguments

`-detail`	Prints a summary with the names of the imported clock-gating modules and the number of instances found for each clock-gating module.
`-hierarchical`	Allows traversing the design hierarchy to process clock-gating instances that were not inserted by Genus. By default, this command only affects instances at the current level of the design hierarchy.
`-keep_name`	Keeps the original clock-gating instance name.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-start_from` *`instance`*

Starts processing clock-gating instances that were not inserted by Genus from the specified hierarchical instance.

By default, the process starts from the current location in the design hierarchy.

`-verbose`

Includes the list of subdesigns that have no imported clock gates in the summary report.

Examples

- The following example shows the minimum information listed when clock-gating instances are successfully imported. The number “2” is the total number of instances imported.

```
legacy_genus:/> clock_gating import -start_from /designs/top -hier
Importing clock_gating logic from /designs/top
Imported 2 Clock Gating instances

2
```

- The following example shows the additional information listed when the `-detail` option is specified.

```
legacy_genus:/> clock_gating import -start_from /designs/top -hier -detail
Importing clock_gating logic from /designs/top

               Detailed report for clock_gating import
 ---------------------------------------------------------------------
                    Module name                     |  Import count
 ---------------------------------------------------------------------
 a_1                                                |    1
 a                                                  |    1
 ---------------------------------------------------------------------

Imported 2 Clock Gating instances

2
```

Related Information

Related command: report clock_gating on page 1256

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating insert_in_netlist

```
clock_gating insert_in_netlist
```

Inserts clock-gating logic in a mapped netlist if the D-input of a flip-flop is driven by a two-input MUX and there is a feedback loop from the Q-output to the D-input through one of the data pins of the two-input MUX.

You should only use this command on a netlist that was already mapped (possibly by a third-party tool).

If you set the `lp_clock_gating_test_signal` attribute before you enter this command, the RC-LP engine can connect the test-control signal to the test pins of the clock-gating logic during clock-gating insertion.

Note: This command allows the flip-flops and MUX logic to be in different hierarchies.

Related Information

Related command:	report clock_gating on page 1256
Affected by this attribute:	lp_clock_gating_test_signal

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating insert_obs

```
clock_gating insert_obs
     [-hierarchical] [-make_obs_module]
     [-max_cg integer]
     [-ignore_clock_constraint]
     [-exclude instance...]
     [-disable_clock -libcell libcell]
```

Inserts and connects circuitry to improve the observability of the design after clock-gating logic is inserted. This command applies to the current design or the current hierarchical instance.

Note: To make sure that the enable signal of the clock-gating logic is observable, set the `lp_clock_gating_add_obs_port` design attribute to `true` before you insert the clock-gating logic.

Observability logic is inserted based on clock information. The clock information is required because only clock-gating logic driven by the same clock can share an observation flip-flop. The clock information can be derived from clock constraints or from the physical connectivity.

Note: This command works on a netlist whose clock-gating logic was either inserted or imported by the Genus-LP engine.

Options and Arguments

`-disable_clock`	Specifies to gate the clock of the observability flip-flops. **Note:** One gating cell is inserted per flip-flop. The Genus-LP engine creates a separate subdesign for each gating cell.
`-exclude` *`instance`*	Prevents insertion of observability logic in the specified hierarchical instances.
`-hierarchical`	Allows insertion and connection of observability logic in the current level of the hierarchy and all its children. By default, this command only affects the current level of the hierarchy.
`-ignore_clock_constraint`	Inserts observability logic based on physical connectivity. By default, observability logic is inserted based on clock constraints defined with the `define_clock` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-libcell` *`libcell`*	Specifies the name of a library cell to be used for gating. **Note:** You can only specify an AND cell to gate the observability logic.
`-make_obs_module`	Creates a separate hierarchy (module) for each observation flip-flop and its associated XOR tree.
`-max_cg` *`integer`*	Specifies the maximum number of clock-gating cells that can be observed per observation flip-flop. Specify an integer between 1 and 32. *Default*: `8`

Related Information

Related commands:	define_clock on page 456
	report clock_gating on page 1256
Affected by this attribute:	lp_clock_gating_add_obs_port

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating join

```
clock_gating join
     [-hierarchical] [-max_level integer]
     [-multi_fanouts] [-start_from instance]
```

Combines multiple stages of clock-gating logic into a single clock gating instance with a complex enable function.

Note: This command works on a netlist whose clock-gating logic was either inserted or imported by the Genus-LP engine.

Options and Arguments

`-hierarchical`	Allows joining of clock-gating logic down the hierarchy starting from the current directory. By default, this command only affects the current level of the hierarchy.
`-max_level` *`integer`*	Specifies the maximum levels of clock-gating instances that can be combined. If you specify n, n+1 stages can be combined. *Default*: `1` allowing 2 levels to be combined.
`-multi_fanouts`	Allows joining even if the root stage clock-gating instance is driving multiple clock-gating instances.
`-start_from` *`instance`*	Joins the clock-gating logic starting from the specified hierarchical instance.

Examples

- The following command allows joining clock-gating instances across the hierarchy of hierarchical instance `i1`.

  ```
  clock_gating join -hierarchical -start_from [find / -inst i1]
  ```

- The following command allows joining three stages of clock-gating instances across the hierarchy starting from the current directory.

  ```
  clock_gating join -hierarchical -max_level 2
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: report clock_gating on page 1256

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating rebalance

```
clock_gating rebalance
     [-strict_min_flop_constraint]
```

Regroups the flops of the clock-gating instances driven by the same inputs based on the values of the `lp_clock_gating_min_flop` and `lp_clock_gating_max_flops` attributes. As a result, the number of clock-gating instances may decrease or increase.

You can use this command after generic synthesis (when clock-gating instances are inserted) or after importing RTL clock-gating instances.

Rebalancing is performed on the full design, and on clock-gating instances inserted by Genus.

Rebalancing is done on clock-gating instances that

- are within the same hierarchy
- are not marked *preserved*
- have the same enable signal
- have the same timing exceptions
- have the same type

Options and Arguments

`-strict_min_flop_constraint`

Requests to strictly follow the minimum flop constraint specified by the `lp_clock_gating_min_flop` attribute.

Related Information

Related commands:	clock_gating import on page 1228
	syn_generic on page 526
Related attributes:	lp_clock_gating_max_flops
	lp_clock_gating_min_flops

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating remove

```
clock_gating remove
     [ -hierarchical [-obs_only]
     | -cg_list instance_list [-obs_only]
     | -flops flops]
     [ -no_verbose]
     [ -effort {low|high}]
```

Removes clock-gating logic inserted by the Genus-LP engine from the current design or the current hierarchical instance.

Note: This command works on a netlist whose clock-gating logic was either inserted or imported by the Genus-LP engine.

Options and Arguments

`-cg_list` *`instance_list`*

Specifies a list of clock-gating instances to be removed. Use a full path name to identify these instances.

Note: If you specify a clock-gating instance with an incomplete path, the tool searches for that instance from the root of the design hierarchy and might select multiple instances with the same name from different hierarchies.

`-effort {high|low}`

Specifies the effort level.

Choosing low effort results in better runtime performance but at the cost of an area increase. In this case, the Genus-LP engine reconstructs the original MUX and feedback loop from the flip-flop to the MUX.

For large designs high effort can result in long runtimes, but the feedback logic is optimized.

Default: `low`

`-flops` *`flops`*

Removes clock gating from the specified flops (that is, recreates the feedback loop for those flops).

If you specified all flops that are gated by the same clock-gating instance, the clock-gating instance will also be removed.

`-hierarchical`

Removes all clock-gating logic in the hierarchy of the current design or subdesign.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	By default, this command only affects the current level of the hierarchy.
`-no_verbose`	Suppresses info and warning messages.
`-obs_only`	Removes only the observability logic.

Example

- The following command removes all clock-gating instances in the hierarchy of subdesign `sub1`.

```
legacy_genus:/designs/alu/subdesigns/sub1> clock_gating remove -hier
```

- The following command removes the clock-gating instances `RC_CG_HIER_INST_121` and `RC_CG_HIER_INST_122` from the current design hierarchy.

```
legacy_genus:/> clock_gating remove -cg_list \
/designs/top/instances_hier/RC_CG_HIER_INST_121 \
/designs/top/instances_hier/RC_CG_HIER_INST_122

Clock-gating instance removed   /designs/top/instances_hier/RC_CG_HIER_INST_121
Clock-gating instance removed   /designs/top/instances_hier/RC_CG_HIER_INST_122
```

Related Information

Related command: report clock_gating on page 1256

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating share

```
clock_gating share
     [-hierarchical] [-max_level integer]
     [-max_stage {integer|string}]
```

Extracts the enable function shared by clock-gating logic and inserts shared clock-gating logic with the common enable sub function as the enable signal. The resulting netlist has multiple stages of clock-gating logic.

Note: This command works on a netlist whose clock-gating logic was either inserted or imported by the Genus-LP engine.

Options and Arguments

`-hierarchical`
: Inserts shared clock-gating logic down the hierarchy starting from the design or current hierarchical instance.

  By default, this command only affects the current level of the hierarchy.

`-max_level integer`
: Specifies the maximum levels of logic (buffers and inverters excluded) to traverse in the enable fanin of clock-gating instances to extract the common enable function.

  Default: `5`

`-max_stage {integer|string}`
: Specifies the maximum number of stages of shared clock-gating logic.

  To specify the same maximum number of stages for all clocks, specify an integer.

  To specify the maximum number of stages per clock, use a string. The string must have the following format:

  { {*clock integer*} {*clock integer*} ...}

  If this option is not specified, no limit is applied to the number of stages.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command allows sharing clock-gating logic across the hierarchy starting from the current directory and allows traversing two levels of logic to extract the common enable function.

  ```
  clock_gating share -hierarchical -max_level 2
  ```

- The following command will insert a maximum of 2 stages of shared clock-gating logic for clock `clk1` and a maximum of 3 stages of clock-gating logic for clock `clk2`.

  ```
  clock_gating share -max_stage { {clk1 2} {clk2 3} }
  ```

Related Information

Related command: report clock_gating on page 1256

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating split

```
clock_gating split
     [-hierarchical] [-max_level integer]
     [-power_driven] [-start_from instance]
```

Splits a single clock gating instance with a complex enable function into multiple stages of clock-gating logic.

Note: This command works on a netlist whose clock-gating logic was either inserted or imported by the Genus-LP engine.

Options and Arguments

`-hierarchical`	Allows splitting of clock-gating logic down the hierarchy starting from the current directory. By default, this command only affects the current level of the hierarchy.
`-max_level` *`integer`*	Specifies how many times a complex enable function can be split. *Default*: `1` allowing the complex enable function to be split into two stages.
`-power_driven`	Forces to use the signal with smallest toggle rate as the root-level enable. By default, the Genus-LP engine considers timing first and uses the late signal as the root-level enable.
`-start_from` *`instance`*	Splits the clock-gating logic starting from the specified hierarchical instance.

Examples

- The following command allows splitting clock-gating instances across the hierarchy of hierarchical instance `i1`.

  ```
  clock_gating split -hierarchical -start_from [find / -inst i1]
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command allows splitting a single clock-gating instance with a complex enable function into three stages of clock-gating instances across the hierarchy starting from the current directory.

```
clock_gating split -hierarchical -max_level 2
```

Related Information

Related command: report clock_gating on page 1256

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_saif

```
read_saif [-scale scale_factor]
     [-update [-weight weight_factor]]
     [-verbose] [-ignorecase] [-instance instance] file
```

Reads switching activity information in Synopsys switching activity interchange format (SAIF) and converts it internally to the Toggle Count Format (TCF) for power estimation.

The `read_saif` command can read files that have been compressed with gzip (`.gz` extension). The `.gz` file is unzipped in memory while the file is read in.

Note: If you read in subsequent SAIF files without the `-update` option, only the probability values and toggle counts of the pins and nets in the current SAIF file are overwritten. The other net values remain unchanged.

The following applies when *updating* the probability values and toggle counts:

- If the probability values and toggle rates were not previously user asserted, the updated probability and toggle rates are determined by the values specified in the SAIF file.
- If the probability and toggle rates were previously user asserted, the new probability and toggle rates are calculated as follows:

  ```
  prob_new = (prob_old + w * prob_spec)/(1+w)
  tr_new = (tr_old + w * tc_spec/duration_spec)/(1+w)
  ```

  where `prob_old` and `tr_old` are the stored values, and `prob_spec`, `tc_spec,` and `duration_spec` are the probably, toggle count, and duration values derived from the new SAIF file.

Options and Arguments

file	Specifies the name of the SAIF file. The file can have any name, suffix, or length.
`-ignorecase`	Ignores the case of module, net, and pin names in the SAIF file when searching for the matching module, net, or pin in the design. By default, case is taken into account. **Note:** Using this option might result in increased run time.
`-instance` *instance*	Reads in the switching activities for the specified instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The instance name can refer to an instance in the design loaded in Genus, or can refer to an instance name in the SAIF file.

- If the instance name refers to an instance in the design loaded in Genus, the Genus-LP engine asserts switching activities on that instance in the loaded design.

  In this case, a *complete* design is loaded in Genus. However, the SAIF file is incomplete and contains only switching activities for the specified instance.

- If the instance name refers to an instance in the SAIF file, the Genus-LP engine asserts switching activities on the design loaded in Genus.

  The design loaded in Genus is

  - a *partial* design if the top-level instance in the SAIF file corresponds to the full design, while the *specified* instance is a lower-level instance.
  - the full design if the *specified* instance in the SAIF file corresponds the top-level design scope in the SAIF file.

  In this case, the SAIF file is a hierarchical SAIF and contains switching activities for the full design.

  Note: The name of the instance in the SAIF file does not need to match the name of the design.

`-scale` *`scale_factor`*

Scales the toggle counts in the SAIF file by dividing them by the specified factor. Use a positive (non-zero) floating number.

Default: `1.0`

`-update`

Indicates that you are updating the probability values and toggle counts.

`-verbose`

Prints a message for each net that is asserted.

Default: Silent mode. Prints the percent completion messages.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-weight` *`weight_factor`*

Specifies the relative weight of the probability values and toggle rates in the new SAIF file with respect to the probability values and toggle rates currently stored in the design. Use a positive floating number. This option is only valid with the `-update` option.

Default: `1.0`

Examples

For the following examples, consider the following SAIF file (`and2.saif`):

```
(SAIFILE
(SAIFVERSION "2.0")
(DIRECTION "backward")
(DESIGN "a")
(DATE "date")
(VENDOR "Cadence Design Systems Inc.")
(PROGRAM_NAME "program")
(VERSION "version")
(DIVIDER / )
(TIMESCALE 1 ns)
(DURATION 1000.00)
(INSTANCE a
    (NET
        (in2
            (T0 700) (T1 300) (TC 16)
        )
        ("in1"
            (T0 900) (T1 100) (TC 9)
        )
        (out
            (T0 100) (T1 900) (TC 7)
        )
  )
)
)
```

- The following command reads the SAIF file with the `-verbose` option:

```
legacy_genus:/> read_saif and2.saif -verbose
10.0 % done
...
60.0 % done
  Setting attribute of net 'in2': 'lp_asserted_probability' = 0.30000
  Setting attribute of net 'in2': 'lp_asserted_toggle_rate' = 0.016000
70.0 % done
  Setting attribute of net 'in1': 'lp_asserted_probability' = 0.10000
  Setting attribute of net 'in1': 'lp_asserted_toggle_rate' = 0.009000
80.0 % done
  Setting attribute of net 'out': 'lp_asserted_probability' = 0.90000
  Setting attribute of net 'out': 'lp_asserted_toggle_rate' = 0.007000
90.0 % done
Nets/ports asserted in SAIF file : 3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
Total Nets/ports in SAIF file    : 3
-------------------------------------------------------
Asserted Primary inputs in design              : 2 (100.00%)
Total connected primary inputs in design       : 2 (100.00%)
-------------------------------------------------------
Asserted sequential outputs                    : 0 (0%)
Total connected sequential outputs             : 0 (100.00%)
-------------------------------------------------------
Total nets in design                 : 4 (100.00%)
Nets asserted                        : 3 (75.00%)
Clock nets                           : 0 (0.00%)
Constant nets                        : 0 (0.00%)
Nets with no assertions              : 1 (25.00%)
-------------------------------------------------------
```

- The following command scales the toggle counts in the SAIF file by a factor 2:

  ```
  legacy_genus:/> read_saif and2.saif -scale 2.0
  ```

  Check the asserted toggle rates on nets `in1` and `in2`:

  ```
  legacy_genus:/> get_attr lp_asserted_toggle_rate nets/in1
  0.004500
  legacy_genus:/> get_attr lp_asserted_toggle_rate nets/in2
  0.008000
  ```

- In the following example, assume you have read in the `and2.saif` file, and you read in the following `and2_new.saif` file:

  ```
  (SAIFILE
  ...
  (TIMESCALE 1 ns)
  (DURATION 1000.00)
  (INSTANCE a
      (NET
          (in1
              (T0 900) (T1 100) (TC 5)
          )
      )
  )
  )
  ```

  The following command updates the stored switching activities with the data in the `and2_new.saif and` gives a two times higher weight on the values in the `and2_new.saif` file.

  ```
  read_saif -update -weight 2 and2_new.saif
  ```

  Check the asserted toggle rates on nets `in1`:

  ```
  legacy_genus:/> get_attr lp_asserted_toggle_rate nets/in1
  0.006333
  ```

  This can be calculated as follows:

  ```
  tr_new = (tr_old + w * tc_spec/duration_new)/(1+w)
         = (0.009000 + 2 * 0.005000)/ (1+2) = 0.006333
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affects this command:	report power on page 1257
Related command:	write_saif on page 1263
Sets these attributes:	lp_asserted_probability
	lp_asserted_toggle_rate
Related attributes:	lp_probability_type
	lp_toggle_rate_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_tcf

```
read_tcf [-scale scale_factor]
     [-update [-weight weight_factor]]
     [-instance instance]
     [-tcf_instance instance]
     [-ignorecase] [-verbose] file
```

Reads or updates probability values and toggle counts of the pins and nets in the specified Toggle Count Format (TCF) file and stores the assertions as pin or net attributes, so they can be used for power estimation and optimization.

The `read_tcf` command can read files that have been compressed with gzip (`.gz` extension). The `.gz` file is unzipped in memory while the file is read in.

Note: If you read in subsequent TCF files without the `-update` option, only the probability values and toggle counts of the pins and nets in the current TCF file are overwritten. The other net values remain unchanged.

When *updating* the probability values and toggle counts, the new probability and toggle rates are calculated as follows:

```
prob_new = (prob_old + w * prob_spec)/(1+w)
tr_new = (tr_old + w * tc_spec/duration_spec)/(1+w)
```

where

- `prob_old` and `tr_old` are either the user-asserted values or the values computed using the power simulation engine.
- `prob_spec`, `tc_spec`, and `duration_spec` are the probability, toggle count, and duration values specified in the new TCF file.

Important

You should not execute any command (such as `change_names` or `ungroup`) that can cause changes in the name of the design objects before you read the TCF file. Otherwise, the `read_tcf` command may not find some design objects.

Options and Arguments

file	Specifies the name of the TCF file. The file can have any name, suffix, or length.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-ignorecase`

Ignores the case of module, net, and pin names in the TCF file when searching for the matching module, net, or pin in the design.

By default, case is taken into account.

Note: Using this option might result in increased run time.

`-instance` *`instance`*

Specifies the name of an instance in the Genus hierarchy to which the parsed TCF hierarchy (specified through the `-tcf_instance` option) corresponds.

For example, if a *partial* design is loaded in Genus but you have a TCF file that contains switching activities for the full design, the top design in Genus will correspond to an instance in the TCF hierarchy.

You can also have a full design loaded in Genus, but only have a partial TCF. In that case you need to specify the name of the instance in the Genus hierarchy to which the TCF file applies.

By default, the TCF file applies to the top design in the Genus hierarchy. If multiple top designs exists, you must specify the name of the top design.

`-scale` *`scale_factor`*

Scales the toggle counts in the TCF file by dividing them by the specified factor. Use a positive (non-zero) floating number.

Default: `1.0`

`-tcf_instance` *`instance`*

Starts parsing the TCF hierarchy from the specified instance.You can specify to start parsing from the top or from a particular instance in the TCF hierarchy.

Default: first instance encountered is used as the top instance.

`-update` Indicates to update the probability values and toggle counts.

`-verbose` Prints a message for each net that is asserted.

Default: Silent mode. Prints the percent completion messages.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-weight` *`weight_factor`*

Specifies the relative weight of the probability values and toggle rates in the new TCF file with respect to the probability values and toggle rates currently stored in the design. Use a positive floating number. This option is only valid with the `-update`.

Default: `1.0`

Examples

- Consider the following TCF file (`example1.tcf`):

```
tcffile () {
   tcfversion   :       "1.0";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   instance () {
     pin () {
         "i_12/Z"       :       "0.566 747";
         "n_n1/B"       :       "0.516 475";
         "hier1/i_0/Z"  :       "0.5 500";
         "hier1/n_n0/Z" :       "0.5 500";
         "hier1/n_n0/A" :       "0.5 500";
         "hier1/i_0/A"  :       "0.5 500";
         "hier1/i_0/B"  :       "0.5 500";
         "n_n1/A"       :       "0.61 516";
      }
    }
 }
```

To read this TCF file (`example1.tcf`), use the following command:

```
legacy_genus:/> read_tcf example1.tcf
```

- In the following TCF file (`example2.tcf`), the only difference with the previous TCF file is that the duration in `example2.tcf` is half of the duration in `example1.tcf`.

```
tcffile () {
   tcfversion   :       "1.0";
   duration     :       "0.75000e+05";
   unit         :       "ns";
   instance () {
     pin () {
         "i_12/Z"       :       "0.566 747";
         "n_n1/B"       :       "0.516 475";
         "hier1/i_0/Z"  :       "0.5 500";
         "hier1/n_n0/Z" :       "0.5 500";
         "hier1/n_n0/A" :       "0.5 500";
         "hier1/i_0/A"  :       "0.5 500";
         "hier1/i_0/B"  :       "0.5 500";
         "n_n1/A"       :       "0.61 516";
      }
    }
 }
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

To make the toggle rates on all pins the same as in the previous example, use the following command:

```
legacy_genus:/> read_tcf -scale 2.0 example2.tcf
```

- Consider the following TCF file (`example1.tcf`):

```
tcffile () {
   tcfversion   :       "1.0";
   duration     :       "1.000000e+05";
   unit         :       "ns";
   instance () {
     pin () {
         "i_0/A"        :       "0.5 500";
         "i_0/B"        :       "0.6 600";
         "i_0/Z"        :       "0.7 700";
      }
    }
 }
```

Assume you read this TCF file with the following command:

```
legacy_genus:/> read_tcf example1.tcf
```

Now consider the following TCF file (`example2.tcf`):

```
tcffile () {
   tcfversion   :       "1.0";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   instance () {
     pin () {
        "i_0/A"        :       "0.5 600";
        "i_0/B"        :       "0.6 750";
        "i_0/Z"        :       "0.7 900";
      }
    }
 }
```

Assume you read this TCF file with the following command:

```
legacy_genus:/> read_tcf -update -weight 0.5 example2.tcf
```

You would get the same result by:

a. Creating the following TCF file (`example3.tcf`)

```
tcffile () {
   tcfversion   :       "1.0";
   duration     :       "1.500000e+05";
   unit         :       "ns";
   instance () {
     pin () {
        "i_0/A"        :       "0.5 700";
        "i_0/B"        :       "0.6 850";
        "i_0/Z"        :       "0.7 1000";
      }
    }
 }
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

b. Reading the `example3.tcf` file using the following command:

```
read_tcf example3.tcf
```

Related Information

Affects this command:	report power on page 1257
Sets these attributes:	lp_asserted_probability
	lp_asserted_toggle_rate
Related attributes:	lp_probability_type
	lp_toggle_rate_type

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_vcd

```
read_vcd
     [ -static] [-scale scale_factor]
     [ -activity_profile] [-time_window time]
     [-simvision] [-write_sst2 file]
     [-start_time start_monitoring_time]
     [-end_time end_monitoring_time]
     [-instance instance] [-vcd_scope instance]
     [-ignorecase] vcd_file
```

Reads a Value Change Dump (VCD) file for power analysis. You can

- Perform static power analysis
- Build an activity profile

Note: If no options are specified, static power analysis is performed by default.

The `read_vcd` command can read files that have been compressed with gzip (`.gz` extension). The `.gz` file is unzipped while the file is read in.

Important

You should not execute any command (such as `change_names` or `ungroup`) that can cause changes in the name of the design objects before you read the VCD file. Otherwise, the `read_vcd` command may not find some design objects.

Options and Arguments

`-activity_profile`

Builds a profile of the activities for the set scope without annotating the switching activities to the design.

By default, profiling is done for the whole design. You can limit the scope to a portion of the design by setting the `lp_dynamic_analysis_scope` attribute to `true` on those instances for which you want to build the profile.

The Genus-LP engine captures the toggle count of objects within a given time window. The object can be a net, pin or a hierarchical instance. For a hierarchical instance, the activity will be the sum of the activities of the objects in that instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: By default, the `read_vcd` command performs static power analysis if neither the `-static` or `-activity_profile` option was specified.

`-end_time` *`end_monitoring_time`*

Specifies the time you want to end monitoring the switching activities or events. Specify a value larger than zero in picoseconds.

By default, the activities or events are monitored till the end (last timestamp of the VCD file).

`-ignorecase`

Ignores the case of module, net, and pin names in the VCD file when searching for the matching module, net, or pin in the design.

By default, case is taken into account.

Note: Using this option might result in increased run time.

`-instance` *`instance`*

Specifies the name of the instance in the Genus hierarchy to which the parsed VCD hierarchy (specified through the `-vcd_scope` option) corresponds.

For example, if a *partial* design is loaded in Genus but you have a VCD file that contains switching activities for the full design, the top design in Genus will correspond to an instance in the VCD hierarchy.

You can also have a full design loaded in Genus, but only have a partial VCD. In that case you need to specify the name of the instance in the Genus hierarchy to which the VCD file applies.

By default, the VCD file applies to the top design in the Genus hierarchy. If multiple top designs exists, you must specify the name of the top design.

`-scale` *`scale_factor`*

Scales the toggle counts in the VCD file by dividing them by the specified factor. Use a positive (non-zero) floating number.

Default: `1.0`

`-simvision`

Invokes SimVision to display the activity profile.

Note: To use this option you need to have SimVision installed and your operating system `PATH` environment variable must include the path to SimVision.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-start_time` *`start_monitoring_time`*

Specifies the time you want to start monitoring the switching activities or events. Specify a value larger than zero in picoseconds.

By default, the first timestamp in the VCD file is considered as the start time to monitor.

`-static`

Calculates the switching activities of each of the nets and pins from the time you want to start monitoring the switching activities to the time you want to stop monitoring, and then stores the information as assertions on the nets and pins.

Note: By default, the `read_vcd` command performs static power analysis if neither the `-static` or `-activity_profile` option was specified.

`-time_window` *`window`*

Specifies the time increment, in picoseconds, in which you want the Genus-LP engine to divide the period during which the events are monitored. The specified time window must be larger than zero.

By default, the time window is calculated based on the setting of the `lp_power_analysis_effort` root attribute and the values of the `-start_time` and `-end_time` options.

Important

If the `-start_time` and `-end_time` options are not specified, the time window will correspond to the complete simulation time from the VCD file.

`vcd_file`

Specifies the name of the value change dump (VCD) file.

`-vcd_scope` *`instance`*

Starts parsing the VCD hierarchy from the specified instance. You can specify to start parsing from the top or from a particular instance in the VCD hierarchy.

Default: first scope encountered is used as the top scope

`-write_sst2` *`file`*

Specifies the prefix of the SST2 database files to generate to view the data in other waveform viewers.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following command reads `my_vcd.vcd`, generates an activity profile from 10 to 100 ps based on a time window of 10ps, and invokes SimVision to display the activity profile.

```
read_vcd -vcd_scope mid2 -activity_profile -start_time 10 -end_time 100 \
-time_window 10 -simvision my_vcd.vcd
```

Related Information

Affects this command:	report power on page 1257
Related attributes:	lp_dynamic_analysis_scope
	lp_power_analysis_effort

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report clock_gating

Refer to report clock_gating in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power

Refer to report power in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

state_retention

```
state_retention
     {connect_power_gating_pins | swap}
```

Defines the aspects of mapping to state retention cells.

Options and Arguments

`connect_power_gating_pins`	Connects the power gating pins in a mapped netlist.
`swap`	Replaces sequential cells with their equivalent state retention power gating cells in a mapped netlist.

Related Information

Related commands:	state_retention connect_power_gating_pins on page 1259
	state_retention swap on page 1260

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

state_retention connect_power_gating_pins

```
state_retention connect_power_gating_pins
```

Connects the power gating pins according to the driver specifications.

Use this command if you

- Replaced the sequential cells with state-retention cells after mapping and did not specify to hook up the power gating pins at that time (used `state_retention swap` command without the `-connect_power_gating_pins` option).
- Specified the driver specifications (`state_retention define_driver` commands) after mapping (although the mapping instructions were given before mapping)

Related Information

Related attributes:	power_gating_pin_class
	power_gating_pin_phase

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

state_retention swap

```
state_retention swap
     [-hierarchical]
     [-start_from instance]
     [-connect_power_gating_pins]
```

Replaces sequential cells with their equivalent state retention power gating cells in a mapped netlist.

Options and Arguments

`-connect_power_gating_pins`

Hooks up the power gating pins with their drivers.

The drivers are specified through the `lp_srpg_pg_driver` instance attribute.

`-hierarchical`

Allows traversing the design hierarchy to map.

By default, this command only affects instances at the current level of the design hierarchy.

`-start_from instance`

Starts replacing sequential cells from the specified hierarchical instance.

By default, the process starts from the current location in the design hierarchy.

Related Information

Affected by these attributes: hdl_enable_proc_name

hdl_proc_name

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_forward_saif

```
write_forward_saif
     [-library library_path...
     |-library_domain library_domain]
     [ > file ]
```

Prints the library forward SAIF file. This file contains the state-dependent and path-dependent (SDPD) directives needed to generate backward SAIF files during simulation.

Note: You do not need to have any designs loaded to write out a forward SAIF file. You only need to have the libraries loaded.

Options and Arguments

file
: Specifies the file to which to write the library forward SAIF information.

  If not specified, the information is written to the screen.

-library *library_path...*
: Specifies the paths to the libraries for which to generate the forward SAIF information.

  If not specified, the information is generated for all libraries that are loaded.

-library_domain *library_domain*
: Specifies the path to the library domain containing the libraries for which to generate the forward SAIF information.

  If not specified, the information is generated for all libraries in all library domains.

  Note: This option only applies if you created library domains using the create_library_domain command.

Examples

- The following example redirects the forward SAIF information for the `cg` library to the `my.saif` file:

  ```
  write_forward_saif -library /libraries/cg > my.saif
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example redirects the forward SAIF information for the libraries in library domain `d1` to the screen:

```
write_forward_saif -library_domain /libraries/library_domains/d1
```

Related Information

Related commands:	create_library_domain on page 1282
	read_saif on page 1242

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_saif

```
write_saif [-duration simulation_period] [-computed]
     [-boundary_only] [-include_hier_ports] [> file]
```

Writes a hierarchical SAIF file containing the user-asserted or computed (if requested) probability and toggle count of the pins in the design.

By default, the Genus-LP engine writes out the user-asserted switching activities of all leaf instance output pins and primary inputs ports.

The `write_saif` command writes out a compressed SAIF file if you add the `.gz` extension to the file name, but is not removed from the directory.

Options and Arguments

`-boundary_only`
: Writes out the switching activities of the primary inputs ports and the leaf sequential instance output pins.

`-computed`
: Adds the computed probability and toggle count of the pins and nets to the SAIF file. By default only the asserted values are written out.

`-duration` *`simulation_period`*
: Modifies the duration for which the toggle count is written in the SAIF file. By default, toggle counts are given for a duration of one second. By modifying the duration, smaller toggle counts (numbers) can be written. For example, if the toggle rate is `3e-3/ns`, the default printed toggle count is 300000. If the simulation period is set to`1e+5 ns`, the printed toggle count will be 300.

  Note: Do not choose the duration too small, otherwise the toggle count will be rounded to 0, because only integer numbers are written out.

  Default: `1e+9 ns`

`file`
: Specifies the name of the file to which to write the probability values and toggle count values.

`-include_hier_ports`
: Includes the (computed) switching activities for the hierarchical output ports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example writes out a SAIF file with the user-asserted switching activities.

```
write_saif > my.saif
```

Note: If you did not read in a TCF or SAIF file, and you did not set toggle rate or probability values on any nets, this SAIF file will not contain any switching activities because you did not request to write out the computed values.

Related Information

Affected by these attributes:	lp_asserted_probability
	lp_asserted_toggle_rate
Related command:	read_saif on page 1242

By default, the command writes out the toggle count for a duration of 1s. For example, if the toggle rate is 3e-3/ns and the simulation period is 1e5 ns, the printed toggle count will be 300

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_tcf

```
write_tcf [-duration simulation_period] [-computed]
     [-hierarchical] [-include_hier_ports] [-pin] [> file]
```

Writes a TCF file containing the user-asserted or computed (if requested) probability and toggle count of the pins in the design.

The `write_tcf` command writes out a compressed TCF file if you add the `.gz` extension to the file name.

Options and Arguments

`-boundary_only`
: Writes out the switching activities of the primary inputs ports and the leaf sequential instance output pins.

`-computed`
: Adds the computed probability and toggle count of the pins and nets to the TCF file. By default only the asserted values are written out.

`-duration` *`simulation_period`*
: Modifies the duration for which the toggle count is written in the TCF file. By default, toggle counts are given for a duration of 1s

  By modifying the duration, smaller toggle counts can be written. For example, if the toggle rate is 3e-3/ns, the default printed toggle count is 300000. If the simulation period is 1e5 ns, the printed toggle count will be 300.

  Note: Do not choose the period too small, otherwise the toggle count will be rounded to 0, because only integer numbers are written out.

  Default: `1e+9 ns`

`file`
: Specifies the name of the file to which to write the probability values and toggle count values.

`-hierarchical`
: Writes out a TCF file in hierarchical format.

  By default, the Genus-LP engine writes out a flat TCF file.

`-include_hier_ports`
: Includes the (computed) switching activities for the hierarchical output ports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-pin`	Writes out a pin-based TCF file. This implies that switching activities on all pins are written out. By default, the Genus-LP engine writes out the user-asserted switching activities of all leaf instance output pins and primary inputs ports.

Example

- The following example writes out a flat TCF file with the user-asserted switching activities.

  ```
  write_tcf > my.tcf
  ```

  Note: If you did not read in a TCF or SAIF file, and you did not set toggle rate or probability values on any nets, this TCF file will not contain any switching activities because you did not request to write out the computed values.

Related Information

Related command:	read_tcf on page 1247
Affected by these attributes:	lp_asserted_probability
	lp_asserted_toggle_rate

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

15

Advanced Low Power Synthesis

- apply_power_intent on page 1268
- check_cpf on page 1270
- check_library on page 1272
- check_power_domain on page 1278
- check_power_intent on page 1279
- commit_power_intent on page 1281
- create_library_domain on page 1282
- eval_1801 on page 1283
- optimize_iso_ls on page 1284
- read_power_intent on page 1285
- report low_power_cells on page 1287
- report low_power_intent on page 1288
- reset_power_intent on page 1289
- verify_power_structure on page 1290
- write_power_intent on page 1292

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

apply_power_intent

```
apply_power_intent
     [-design design -module module]
     [-summary]
```

Applies the power intent that was previously read in from power intent file(s).

Note: You should not execute any command (such as `change_names` or `ungroup`) that can cause changes in the name of the design objects before you apply the power intent file(s). Otherwise, the `apply_power_intent` command may not find some design objects.

Options and Arguments

`-design` *`design`*	Specifies the design to which the power intent applies. This option is only required when multiple designs are loaded in the session. If specified, it must be used with the `-module` option.
`-module` *`module`*	Specifies the top module for which to apply the power intent. This option is required when the power intent file was read in before the design is read in, and when the elaborated design name differs from the top module name in the RTL. If specified, it must be used with the `-design` option.
`-summary`	Prints a summary of the power intent. Prints the number of power domains, isolation rules, level shifter rules, state retention rules, and power modes, as well as a list of partially supported commands.

Examples

- Consider the following top module in RTL:

  ```
  module top (in, out, … }
  ..
  endmodule
  ```

  When the design is elaborated, the elaborated design name is now `top_xyz`. In this case you need to map the design name with the corresponding top module name in RTL when applying the power intent.

  ```
  legacy_genus:/>read_power_intent -module top top.1801
  legacy_genus:/>elaborate
  legacy_genus:/>cd des*/*
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/>/designs/top_xyz
legacy_genus:/>apply_power_intent -design top_xyz -module top
```

Related Information

Related commands:

commit_power_intent on page 1281
read_power_intent on page 1285
write_power_intent on page 1292

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_cpf

```
check_cpf  [-design design]
     [-all | -isolation | -level_shifter | -retention]
     [-lp_only] [-detail]
     [-pre_read script] [-pre_exit script]
     [-run_dir directory] [-debug] [-tclmode]
     [-continue_on_error] [-license string] [-generated] [> file]
```

Checks the validity of the CPF rules against the RTL of the design. This enables designers to capture any violations of the low power intent of the design early in the design cycle.

- If no low power rule check errors are detected, the command returns `1`.
- If rule check errors are detected, the command returns `0`. In this case, you need to make the necessary changes to your CPF file before proceeding further with synthesis.

To run this command you need to have access to Conformal ® Low Power.

Options and Arguments

`-all`	Reports all problems with the CPF file. By default, all problems will be reported.
`-continue_on_error`	Allows the tool to continue when low power rule check errors are encountered.
`-debug`	Creates a dofile and other required files for Conformal Low Power allowing you to debug any errors further in Conformal Low Power.
`-design` *`design`*	Specifies the design to which the CPF file applies. This option is only required when multiple designs are loaded in the session.
`-detail`	Provides a detailed report.
`file`	Redirects the report to the specified file.
`-generated`	Checks the validity of the CPF rules against the current state of the design. By default, the tool checks the validity of the CPF rules against the RTL of the design.
`-isolation`	Reports only problems with the isolation rules.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-level_shifter`	Reports only problems with the level-shifter rules.
`-license` *`string`*	Specifies the Conformal license to be used for this command.
`-lp_only`	Reports only the low power rule check errors and warnings. By default, non low-power related issues, such as structural issues are reported as well.
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the CLP `exit` command.
`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-retention`	Reports only problems with the state retention rules.
`-run_dir` *`directory`*	Specifies the directory in which the required files for Conformal Low Power must be stored. *Default*: `.clp`
`-tclmode`	Specifies to generate the dofile as a Tcl script.

Example

The following command reports problems with the level shifter rules.

```
legacy_genus:/designs/top> check_cpf -level_shifter
Using Conformal version xxx.
================================================================
 CPF LEVEL SHIFTER VIOLATIONS
================================================================
CPF_LS1: No level shifter rule specified for power domain crossing.
    Severity: Error      Occurrence: 8

Error   : Low Power rule check did not finish successfully. [RCLP-203] [check_cpf]
        : Fix the errors before proceeding further or set the attribute
'clp_treat_errors_as_warnings' appropriately.
0
```

Related Information

Common Power Format Rule Checks in *Conformal ® Low Power Reference Manual*

Affected by this attribute: wclp_lib_statetable

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_library

```
check_library
     [-isolation_cell] [-level_shifter_cell]
     [-retention_cell]
     [-library_domain library_domain_list]
     [-libcell libcell_list]
     [> file]
```

Allows you to check specific information in the loaded libraries with regards to level shifters, isolation cells, and state retention cells. The report also lists the unusable cells. The information returned can be fine tuned by combining several options.

Without any options specified, this command list the number of level shifters, isolation cells, and state retention cells available in each of the library domains. If no library domains exist, the report lists the library names instead.

Options and Arguments

file
Specifies the name of the file to which to write out the library information.

-isolation_cell
Returns two lists of library cells:

- A list of pure isolation cells with their isolation type
- A list of combo cells with for each cell
  - The isolation type
  - The voltage ranges that they support
  - Whether the combo cell can be used between a lower and higher voltage, or vice versa
  - The valid location for the cell

-level_shifter_cell
Returns a list of level shifter cells found in each of the library domains and specifies for each cell

- The supported input and output range
- Whether the level shifter can be used between a lower and higher voltage, or vice versa
- The valid location for the cell

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-library_domain` *`library_domain_list`*

List the number of level shifters, isolation cells, and state retention cells available in each of the specified library domains.

`-libcell` *`libcell_list`*

If not combined with any other option, indicates for each of the specified library cells to which library domain it belongs, whether it is a level shifter, isolation cell, retention cell, always on cell, and the function of the cell.

`-retention_cell`

Returns for each library domain the following information:

- A list of sequential elements that have no state-retention equivalent
- A list of state-retention cells available in that domain

Examples

- The following command requests a general check of the libraries that were loaded. When requesting a general check, the report lists the number of usable and unusable level shifters, isolation cells, combo cells, and state retention cells in each library or library domain.

```
legacy_genus:/designs/Design2> check_library

============================================================
  ...
  Module:                 Design2
  Library domain:         lib_074v
    Domain index:         0
    Technology libraries: ....
    Operating conditions: _nominal_ (balanced_tree)
  Library domain:         lib_090v
    Domain index:         1
    Technology libraries: ....
    Operating conditions: _nominal_ (balanced_tree)
  Library domain:         lib_110v
    Domain index:         2
    Technology libraries: ....
    Operating conditions: _nominal_ (balanced_tree)
  Library domain:         lib_120v
    Domain index:         3
    Technology libraries: ...
    Operating conditions: _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
================================================================================
Library Domain   Total cells   LS cell   ISO cell   Combo (LS+ISO)   SR Flops
--------------------------------------------------------------------------------
lib_074v(0.74)   345           2         25         2                5
lib_120v(1.2)    341           0         13         0                5
lib_090v(0.9)    345           2         25         2                5
lib_110v(1.1)    343           2         17         2                5
--------------------------------------------------------------------------------

Unusable libcells
=================

================================================================================
Library Domain   Total cells   LS cell   ISO cell   Combo (LS+ISO)   SR Flops
--------------------------------------------------------------------------------
lib_074v(0.74)   249           34        45         16               49
lib_120v(1.2)    232           15        47         8                49
lib_090v(0.9)    249           34        45         16               49
lib_110v(1.1)    239           22        47         10               49
--------------------------------------------------------------------------------
```

- The following command checks the libraries in library domain `lib_120v`.

```
legacy_genus:/> check_library -level_shifter_cell -library_domain lib_120v
============================================================
......
  Library domain:         lib_074v
    Domain index:         0
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Library domain:         lib_120v
    Domain index:         1
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Wireload mode:          enclosed
============================================================


================================================================================
Library Domain   Total cells   LS cell   ISO cell   Combo (LS+ISO)   SR Flops
--------------------------------------------------------------------------------
lib_120v(1.2)    11            3         10         2                0
--------------------------------------------------------------------------------

Unusable libcells
=================

================================================================================
Library Domain   Total cells   LS cell   ISO cell   Combo (LS+ISO)   SR Flops
--------------------------------------------------------------------------------
lib_120v(1.2)    0             0         0          0                0
--------------------------------------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command checks the libraries for isolation cells only. In the example below, the cell names in the Combo cell column were abbreviated for documentation purposes to fit the report. The cell names are not abbreviated by the tool.

```
legacy_genus:/> check_library -isolation_cell
============================================================
......
  Library domain:         lib_074v
    Domain index:         0
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Library domain:         lib_120v
    Domain index:         1
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Wireload mode:          enclosed
============================================================

Pure isolation cells
====================
=========================================================
Library Domain      Isolation Type      Isolation cells
---------------------------------------------------------
lib_074v         enable_high_out_high   OR2XCP
                                        OR2XHP
                                        ....
                                        AND2XCP
                                        AND2XH
                                        ....
lib_120v         enable_high_out_high   OR2XCPPP
                                        OR2XHPPP
                                        ....
                                        AND2XCPPP
                                        AND2XHPPP
                                        ....
---------------------------------------------------------


Combo cells
===========

=========================================================================================
Library   Isolation           Combo          Input       Output        Direction  Location
Domain      type              cell           Range(V)    Range(V)
-----------------------------------------------------------------------------------------
lib_074v enable_high_out_high ......BH1XH    0.74-0.74   1.2-1.2       up        to
                              ......BH1XL    0.74-0.74   1.2-1.2       up        to
         enable_low_out_low   ......1CLXH    1.2-1.2     0.74-0.74     down      to
                              ......1CLXL    1.2-1.2     0.74-0.7      down      to
                              ......PPPAD    0.74-0.74   1.2-1.2       up        to
                              ......PPPAD    0.74-0.74   1.2-1.2       up        to
lib_120v enable_high_out_high ......BH1XH    0.9-0.9     1.2-1.2       up        to
         enable_low_out_low   ......1CLXH    0.9-0.9     1.2-1.2       up        to
-----------------------------------------------------------------------------------------
```

Note: If the libraries would have unusable isolation cells, the report would list the names of the isolation cells per library or library domain.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command checks the function of library cell `LSHLL1CLXL`.

```
legacy_genus:/> check_library -libcell LSHLL1CLXL

==============================================================================
 Library      Libcell      Level   Isolation  Retention  Always   Function
 domain                    shifter  cell        flop       ON
------------------------------------------------------------------------------
lib_074v      LSHLL1CLXL    true    true       false      false    Y = A * B
------------------------------------------------------------------------------
```

- The following command checks the level shifter characteristics of the specified cell.

```
legacy_genus:/> check_library -libcell LSHLL1CLXL -level_shifter_cell
============================================================
......
  Library domain:         lib_074v
    Domain index:         0
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Library domain:         lib_120v
    Domain index:         1
    Technology libraries: ...
    Operating conditions: BALANC_TREE (balanced_tree)
  Wireload mode:          enclosed
============================================================


Level shifter report
====================

=========================================================
 Library     Level shifter       Input      Output    Direction   Location
 Domain           cell         Range(V)    Range(V)
-------------------------------------------------------------------------
lib_074v     LSHLL1CLXL         1.2-1.2    0.74-0.74   down        to
-------------------------------------------------------------------------
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command checks the libraries for state retention cells. The report distinguishes between flip-flops and latches. In the example below the library has no latches.

```
    legacy_genus:/> check_library -retention_cell
============================================================
......
============================================================

Flops without corresponding state-retention flops
-------------------------------------------------

==========================================================================
Library Domain                              Flops
--------------------------------------------------------------------------
110_lib          HD65_LS_DFPHQNX5     HD65_LS_DFPRQNX10    HD65_LS_SDFNRX5
                 HD65_LS_SDFPHQNX10   HD65_LS_SDFPSQNX20
--------------------------------------------------------------------------
Usable state-retention flops
----------------------------------------------------


=============================================================================================
Library Domain                                        Flops
---------------------------------------------------------------------------------------------
090_lib         HD65_LS_SDFNRX10_SRPG     HD65_LS_SDFNRX5_SRPG
110_lib         HD65_LS_SDFPHQNX10_SRPG   HD65_LS_SDFPHQX10_SRPG    HD65_LS_SDFPQNX10_SRPG
                HD65_LS_SDFPQX10_SRPG     HD65_LS_SDFPRQNX10_SRPG   HD65_LS_SDFPRQX10_SRPG
                HD65_LS_SDFPRSQX10_SRPG   HD65_LS_SDFPSQNX10_SRPG   HD65_LS_SDFPSQX10_SRPG
---------------------------------------------------------------------------------------------

Latches without corresponding state-retention latches
-----------------------------------------------------
=============================================================================================
Library Domain                                      Latches
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------

Usable state-retention latches
------------------------------------------------------
=============================================================================================
Library Domain                                      Latches
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
```

Related Information

Related commands: read_power_intent on page 1285

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_power_domain

```
check_power_domain [-design design]
     [-always_on] [-back2back] [-retention]
     [-nets_missing_iso] [-nets_missing_shifter]
     [-bias] [-show_all_crossings]
```

Verifies whether the low power cells in the design conform to the rules and specifications in the loaded CPF file. Specifically, Genus will flag if there are any missing isolation, level shifter, or state retention cells or if the low power cells are not connected appropriately.

Options and Arguments

`-always_on`	Reports missing and unnecessary always-on cells.
`-back2back`	Reports back to back isolation and level-shifter cells in each crossing.
`-bias`	Reports mismatching cells inserted in case of shared nwell.
`-design` *design*	Specifies the name of the design in case of multiple top modules.
`-nets_missing_iso`	Reports missing isolation cells only.
`-nets_missing_shifter`	Reports missing level-shifter cells only.
`-retention`	Reports missing state retention cells only.
`-show_all_crossings`	Reports all crossings without insertion.

Related Information

Related command: commit_power_intent on page 1281

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

check_power_intent

```
check_power_intent [-design design]
     [-always_on] [-retention] [-show_full_names]
     [-isolation] [-level_shifter]
     [-detail]
```

Checks specific power intent issues. After reading the CPF/UPF file, check the validity of the CPF/UPF rules against the RTL of your design. This enables you to capture any violations of the low power intent of the design early in the design cycle. This verifies whether the low power cells in the power domain conform to the rules and specifications of the power intent. Specifically, Genus will flag if there are any missing isolation, level shifter, or state retention cells or if the low power cells are not connected appropriately.

If rule check errors are detected, you need to make the necessary changes to your CPF/UPF file before proceeding further with synthesis.Such CPF/UPF quality checking ensures that the specified power intent is correct and complete.

Options and Arguments

`-always_on`	Reports always-on cell-related issues.
`-design` *design*	Specifies the name of the design in case of multiple top modules.
`-detail`	Show detailed information.
`-isolation`	Reports isolation-related issues.
`-level_shifter`	Reports level-shifter cell-related issues.
`-retention`	Reports state retention cell-related issues.
`-show_full_names`	Specifies to print full names in the report.

Examples

- The following command reports problems with the level shifter rules:

```
check_power_intent -level_shifter
Summary
================================================
Number of pins missing level shifter cell : 860
================================================
Total number of issues found : 860
```

- The following command reports there are no issues with the isolation cells rules:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
check_power_intent -isolation
Total number of issues found : 0
```

Related Information

Related commands: check_power_domain on page 1278
verify_power_structure on page 1290

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

commit_power_intent

```
commit_power_intent
     [-design design]
```

Inserts level-shifter logic and isolation logic as requested based on the rules specified in previously read in power intent file(s).

Options and Arguments

`-design` *`design`*	Specifies the design to which the power intent applies. This option is only required when multiple designs are loaded in the session.

Related Information

Related commands:	apply_power_intent on page 1268 read_power_intent on page 1285 write_power_intent on page 1292

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_library_domain

`create_library_domain` *`domain_list`*

Creates the specified library domains. To use dedicated libraries with portions of the design, you must use this command before you read in any libraries for the specified library domains. The command returns the directory path to the library domains that it creates.

You can find the objects created by the `create_library_domain` command in:

`/libraries/library_domains`

Note: There is no limitation on the number of library domains you can create.

Options and Arguments

`domain_list`	Specifies the names of the library domains to be created. Specify the library domains as a Tcl list.

Examples

- The following example creates three library domains:

```
legacy_genus:/> create_library_domain {dom_1 dom_2 dom_3}
/libraries/library_domains/dom_1 /libraries/library_domains/dom_2 /libraries/
library_domains/dom_3
```

Related Information

Related attribute: library

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

eval_1801

```
eval_1801
     upf_command...
```

Specifies an 1801 (UPF) command to be updated incrementally.

Options and Arguments

upf_command	Specifies the UPF command to be updated.

Example

```
eval_1801 set_isolation ISO_Test -domain PD_Test -applies_to inputs ...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

optimize_iso_ls

```
optimize_iso_ls
     [-design design]
```

Removes any redundant level-shifter cells and isolation cells together with the connected supply nets.

Options and Arguments

`-design` *`design`*	Specifies the design from which to remove the redundant isolation cells and level-shifter cells. This option is only required when multiple designs are loaded in the session.

Related Information

Related command:	commit_power_intent on page 1281

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

read_power_intent

```
read_power_intent
      file [file]...
     [-module module] [-1801 | -cpf] [-version string]
```

Reads the power intent for the module from the specified file(s).
In general, the `read_power_intent` command does not change the netlist.

Options and Arguments

`[-1801 | -cpf]`	Specifies in which format the power intent file are written: the IEEE 1801(UPF) or the CPF format. Currently only the IEEE 1801-2009 version of the UPF standard is supported. *Default:* `-1801`
`file`	Specifies the name of the power intent file. The file can have any name, suffix, or length.
`-module` *`module`*	Specifies the top module for which the power intent is read. This option is required in the following cases: ■ The file is read in before the design is read in. ■ The file is read after the design is loaded, but multiple designs were loaded in the session. By default, the tool assumes that the power content applies to the (single) design that was read in.
`-version` *`string`*	Specifies the version of the power intent file. in UPF format Valid values are: `1.0`, `2.0`, and `2.1`. This option is only used if the UPF file is missing the `upf_version` command.

Related Information

Related commands: apply_power_intent on page 1268

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

commit_power_intent on page 1281

write_power_intent on page 1292

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report low_power_cells

For more information, refer to report power_intent_instances in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report low_power_intent

For more information, refer to report power_intent in Chapter 11, “Analysis and Report.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_power_intent

```
reset_power_intent
     [-design design]
```

Clears the power intent from the design.

Note: This command will not undo any netlist modifications made with the previous power intent.

Options and Arguments

`-design` *`design`*	Specifies the design from which to remove the power intent constraints. This option is only required when multiple designs are loaded in the session.

Related Information

Related commands:	check_cpf on page 1270 read_power_intent on page 1285

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

verify_power_structure

```
verify_power_structure [-design design]
     [-isolation] [-level_shifter] [-retention]
     [-all] [-lp_only] [-nwell_analysis]
     {-pre_synthesis | -post_synthesis | -post_pg}
     [-pre_read script] [-pre_exit script] [-detail]
     [-run_dir directory] [-debug] [-tclmode]
     [continue_on_error] [-license string] [> file]
```

Verifies whether the low power cells in the design conform to the rules and specifications in the loaded CPF file. Specifically, Genus will flag if there are any missing isolation, level shifter, or state retention cells or if the low power cells are not connected appropriately.

To run this command you need to have access to Conformal ® Low Power.

Options and Arguments

`-all`	Reports all violations.
`-continue_on_error`	Allows the tool to continue when low power rule check errors are encountered.
`-design` *`design`*	Specifies the design for which to verify the low power cells. This option is only required when multiple designs are loaded in the session.
`-detail`	Provides a detailed report.
`-debug`	Creates a dofile and other required files for Conformal Low Power allowing you to debug any errors further in Conformal Low Power.
`file`	Redirects the report to the specified file.
`-isolation`	Reports only isolation related violations. **Note:** Has priority over `-all` option.
`-level_shifter`	Reports only level-shifter related violations. **Note:** Has priority over `-all` option.
`-license` *`string`*	Specifies the Conformal license to be used for this command.
`-lp_only`	Reports only the low power rule check errors and warnings. **Note:** This is now the default.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-nwell_analysis`	Checks the design for shared and insulated nwell violations.
`-post_pg`	Checks the design for power and ground connections after mapping.
`-post_synthesis`	Runs Conformal Low Power, after synthesis, on low power cells.
`-pre_exit` *`string`*	Specifies the name of the dofile (script) that must be sourced before the `exit` command.
`-pre_read` *`string`*	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
`-pre_synthesis`	Runs Conformal Low Power, before synthesis, to check the existing low power cells.
`-retention`	Reports only state retention related violations. **Note:** Has priority over `-all` option.
`-run_dir` *`directory`*	Specifies the directory in which the required files for Conformal Low Power must be stored. *Default*: .
`-tclmode`	Specifies to generate the dofile as a Tcl script.

Related Information

Related command:	commit_power_intent on page 1281
Affected by this attribute:	wclp_lib_statetable

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_power_intent

```
write_power_intent [-1801 | -cpf] [-force]
     [-base_name string] [-design design]
     [-overwrite] [-to_macro] [-partition_assemble]
```

Design objects that are specified in he power intent can be renamed. during synthesis, for example during hierarchy ungrouping or user applied naming rules. This command writes out an updated power intent file reflecting the changes in either the IEEE 1801-2009 standard or in CPF format. The command returns the path to the file with the power intent information.

The updated power intent can be formally verified against the original in the Conformal Low Power tool using the compare power intent command.

Note: This command has currently some limitations:

- All unsupported commands and commands that are not applicable to synthesis are written out as they were entered.
- All unsupported options of supported commands are skipped.
- If there are any errors or warnings given during `read_power_intent` or `apply_power_intent`, the power intent written out may not be same as the power intent read.
- If wildcards are used in net or port names specified with a command option, the result may not be as expected.

Options and Arguments

`[-1801 | -cpf]`	Specifies in which format the power intent file must be written: the IEEE 1801-2009 standard, or the CPF format *Default:* `-1801`
`-base_name` *`string`*	Specifies the path and base name for the generated file.
`-design` *`design`*	Specifies the design for which you want to write out the power intent. If you omit this option, the constraints will be reapplied to the current design.
`-force`	Writes out UPF 1.0 in IEEE 1801 format.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

By default, when the input is UPF 1.0, the `write_power_intent` command does not write anything.

`-overwrite`
Allows to overwrite any existing files.

`-partition_assemble`

Specifies to create a power intent file for the design skeleton (design without partitions).

Note: You should only use this option in a CPF partitioning flow after the partitioning is done.

Important

This option is only available as a limited access feature.

`-to_macro`
Creates a CPF macro for the design.

You can only create a CPF macro for the top-level design after you have synthesized the design and committed the CPF file.

You must define the isolation and level-shifter rules for the *input* ports using the `-to` *`domain`* and `-pins` *`pins`* options.

You must define the isolation and level-shifter rules for the *output* ports using the `-from` *`domain`* and `-pins` *`pins`* options.

In addition, you should have checked the quality of your original CPF file after reading the CPF file, performed a low power equivalence check after committing the CPF file, and the results should be clean in both cases.

Related Information

Affected by this attribute:	part_power_intent_file
Related commands:	apply_power_intent on page 1268
	commit_power_intent on page 1281
	read_power_intent on page 1285

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

16

Design Manipulation

- bitblast_all_ports on page 1297
- bitblast_ports on page 1298
- change_link on page 1299
- change_names on page 1301
- clock_gating on page 1308
- connect on page 1309
- clock_gating on page 1308
- connect on page 1309
- create_design on page 1310
- create_hport_bus on page 1311
- create_inst on page 1312
- create_primitive on page 1314
- dedicate_module on page 1317
- delete_unloaded_undriven on page 1318
- disconnect on page 1319
- edit_netlist on page 1320
- edit_netlist bitblast_all_ports on page 1322
- edit_netlist bitblast_port on page 1323
- edit_netlist connect on page 1324
- edit_netlist dedicate_subdesign on page 1326
- edit_netlist delete on page 1328
- edit_netlist disconnect on page 1329

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- edit_netlist flatten_complex_ports on page 1331
- edit_netlist group on page 1332
- edit_netlist hier_connect on page 1334
- edit_netlist new_design on page 1335
- edit_netlist new_instance on page 1336
- edit_netlist new_port_bus on page 1338
- edit_netlist new_primitive on page 1339
- edit_netlist new_subport_bus on page 1341
- edit_netlist ungroup on page 1342
- edit_netlist uniquify on page 1343
- flatten_complex_ports on page 1344
- group on page 1345
- insert_tiehilo_cells on page 1346
- mv on page 1349
- remove_cdn_loop_breaker on page 1351
- reset_design on page 1353
- rm on page 1355
- ungroup on page 1357
- uniquify on page 1359

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

bitblast_all_ports

`bitblast_all_ports {design|subdesign}...`

Bitblasts all ports of the specified design or subdesign. This command is available after elaboration. The name of the bitblasted ports will follow the nomenclature specified by the `bit_blasted_port_style` attribute. The default style is:
`%s_%d` - this means that the individual ports will be named *name_number*

This command will blast all ports of the specified design or subdesign. To blast ports selectively, use `bitblast_ports`.

Options and Arguments

design | *subdesign*	Specifies the design or subdesign in which the ports should be bitblasted.

Related Information

Related command:	bitblast_ports on page 1298
Affected by this attribute:	bit_blasted_port_style
	write_vlog_bit_blast_bus_connections
	write_vlog_bit_blast_constants
	write_vlog_bit_blast_mapped_ports
	write_vlog_bit_blast_tech_cell

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

bitblast_ports

```
bitblast_ports
     {port_bus | subport_bus}
     {design | instance}
```

Bitblasts the specified port_bus (or subport_bus) of the specified design (or hierarchical instance). This command is available after elaboration. The name of the bitblasted ports will follow the nomenclature specified by the `bit_blasted_port_style` attribute. The default style is:
`%s_%d` - this means that the individual ports will be named *`name_number`*

This command will blast only the selected ports on the design or subdesign. To blast all ports, use `bitblast_all_ports`

This command operates on port_bus or subport_bus objects, not on port or pin objects. If a port or a pin is specified, an error will be issued. The parameters must be full Genus database virtual directory (vdir) names.

Options and Arguments

design	Specifies the design to which the port_bus belongs.
instance	Specifies the hierarchical instance to which the subport_bus belongs.
{*port_bus*|*subport_bus*}	Specifies the name of the port_bus or subport_bus to be bit-blasted.

Related Information
/

Related command:	bitblast_all_ports on page 1297
Affected by this attribute:	bit_blasted_port_style

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

change_link

```
change_link -instances instance_list
     { -design_name {instance | subdesign | design}
     | -libcell libcell }
     [-pin_map string] [-lenient]
     [-no_name_change]
     [-change_in_non_uniq_subdesign]
     [-copy_attributes] [-retain_exceptions]
```

Changes the reference of a hierarchical instance to the specified subdesign or design. The command also supports libcell to libcell reference changes as well as a hierarchical instance to libcell changes.

Options and Arguments

`-change_in_non_uniq_subdesign`
: Changes the link of the instance(s) in all instantiations of a non-uniquified subdesign.

`-copy_attributes`
: Copies the attributes of the original leaf instance(s) to the new leaf instances.

`-design_name` {*instance* | *subdesign* | *design*}
: Specifies the design, subdesign, or hierarchical instance to which the link has to be changed.

`-instances` *instance_list*
: Specifies the instance(s) whose reference has to be changed.

`-lenient`
: Leaves the pins floating if a pin map is not found.

`-libcell` *libcell*
: Specifies the library cell to which the instance link has to be changed.

`-no_name_change`
: Prevents renaming of the subdesign name.

`-pin_map` *string*
: Specifies, as a Tcl list of lists, the required pin mapping for swapping.

`-retain_exceptions`
: Keeps the original exceptions of the instance after replacing it with another link.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- The following command changes the reference of the hierarchical instances `top/A`, `top/B`, and `top/C` to `patch`.

```
change_link -instances {/designs/top/instances_hier/A \
/designs/top/instances_hier/B /designs/top/instances_hier/C} \
-design_name /designs/patch
```

- The following example changes the reference of the hierarchical instance `add_0` to the design `add`:

```
legacy_genus:/> change_link -design_name add \
    -instance /designs/test/instances_hier/add_0

CHLNK INFO : Changing link of instance /designs/test/instances_hier/add_0 to
design /designs/add
Warning : Uniquifying instance /designs/test/instances_hier/add_0. New
subdesign is add_1
```

- In the following example, `A1`, `A2`, and `A3` are instances of subdesign `A` which is not uniquified. The following command changes a leaf instance in all instances of subdesign `A`.

```
change_link -instances {/designs/top/instances_hier/A1/instances_comb/g1}
-libcell buf1 -change_in_non_uniq_subd
```

  This command changes not only leaf instance `A1/g1` but also `A2/g1` and `A3/g1` with `buf1`.

  Note: If you omit the `-change_in_non_uniq_subd` option, the tool will issue an error.

- The following command replaces hierarchical instance `A1` with `patch` and tries to retain all exceptions of `A1`.

```
change_link -instances {/designs/top/instances_hier/A1} \
-design_name /designs/patch -retain_exceptions
```

  Exceptions defined for instance `A1` are copied to `patch` if the object for which the exception was originally defined is also found in `patch`. For example, an exception defined on `A1/d_reg` will be retained if `d_reg` also exists in `patch`.

- The following command specifies how to map the pins of the hierarchical instance `A` to the pins of design `new`. Instance `A` has pins `a`, `b`, `c`, and `d`. Design `new` has pins `e`, `f`, `g`, and `h`.

```
change_link -instances /designs/top/instances_hier/A \
-design_name /designs/new -pin_map {{a e} {b f} {c g} {d h}}
```

- The following commands replace hierarchical instance `U1` with top-level design `digit` and hierarchical instance `U2` with top-level design `digit_1` while retaining the original top-level design names as the subdesign names.

```
change_link -instances /des*/top/instances_hier/U1 -design_name /designs/digit
change_link -instances /des*/top/instances_hier/U2 -design_name /des*/digit_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

change_names

```
change_names [ -net | -instance | -design | -subdesign
     | -port_bus | -subport_bus]...
     [-local] [-force] [-lec]
     [-vhdl] [-verilog] [-system_verilog]
     [-prefix string [-name_collision]]
     [-suffix string [-name_collision]]
     [-first_restricted string] [-restricted string]
     [-last_restricted string] [-replace_str string]
     [-reserved_words string] [-max_length number]
     [-map string] [-allowed string... [-regexp]]
     [-check_internal_net_name] [-collapse_name_space]
     [-dummy_net_prefix string] [-skip_bus_net]
     [-case_insensitive] [-lowertoupper] [-uppertolower]
     [-log_changes file [-append_log]] [hier_instance | design]
```

Changes names of nets, busses, instances, designs, subdesigns, ports, port buses, and subport buses. You can specify one of more object types. If no object type is specified, the requested change applies to all object types unless otherwise specified. There is no restriction on the length of the name.

By default, all changes are global: changes are made to all objects (instances of the specified object types) in the design. You can specify the name of a hierarchical instance to restrict the changes to only the objects in that instance.

To change the name of a single object (net, instance, and so on), use the `mv` command.

Note: This command does not affect ILMs, including the name of its module, instances, subports, and nets.

Options and Arguments

`-allowed` *`string`*	Specifies the characters that are allowed in names. Any characters that are not in the allowed list will be ignored in the resulting names. The minimum specification is 10 characters. To allow all the letters from a to z in capital and lower case letters, you must specify all of them. That is, you cannot use a dash (“-”) to indicate inclusion.
`-append_log`	Appends the information of the last `change_names` command to the logfile specified with the `-log_changes` option

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If you omit this option, the information of the last `change_names` command overwrites the current information in the logfile.

Note: You can only specify this option if you specified the `-log_changes` option.

`-case_insensitive` Does not take case sensitivity into account.

`-check_internal_net_name`

Adds the suffix `_int` to any net whose name matches that of a port or subport but is not connected to that port or subport.

`-collapse_name_space`

Adds the suffix `_design` to either the port, subport, net, or subdesign in a hierarchy only if they have similar Verilog names.

`-design` Changes the names of design objects.

`-dummy_net_prefix` *`string`*

Specifies the prefix to use for the names of dummy nets when writing out the netlist or HDL.

Note: This option does not change the names in the design hierarchy.

`-first_restricted` *`string`*

Specifies the characters that cannot be used as first character in a name.

`-force` Forces the name change even if the object name is preserved.

`hier_instance` | *`design`*

Specifies the name of the hierarchical instance or the design to which the changes must be applied.

`-instance` Changes the names of instance objects.

`-last_restricted` *`string`*

Specifies the characters that cannot be used as last character in a name.

`-lec` Captures the changes in the log file in LEC preferred format.

`-local` Restricts the changes to the current directory.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

To restrict the changes to the top-level of the design, specify:

`-local [find / -design design]`

Default: global changes

`-log_changes file`

Specifies the name of the logfile that shows which names were changed using the `change_names` command and the result of the changes.

`-lowertoupper`

Changes the names of all objects from lowercase to uppercase. This applies to objects of type instance, port, net, subport, design, and subdesign.

`-map {{"from" "to"}...}`

Maps the specified *from* string to the specified *to* string.

Enclose each string in double quotes and separate the strings with a space. Enclose each set in braces. If you specify several sets, separate them with spaces and enclose the list of all sets with braces.

`-max_length integer`

Limits the length of the changed name to the specified number. If the resulting names are longer than the specified integer, the last letters will be truncated.

`-name_collision`

Indicates to only use the specified prefix or suffix values to change object names if a name clash would occur while executing the `change_names` command using other options.

`-net`

Changes the names of net objects.

`-port_bus`

Changes the names of the top-level port bus objects.

Note: You cannot change the left bracket, "[", and the right bracket, "]", because they are a part of the bus name when referencing individual bits of the bus.

`-prefix string`

Adds a prefix to the names of the objects to be changed.

`-regexp`

Allows you to specify character ranges with the `-allowed` option or `-first_restricted` option.

`-replace_str string`

Specifies the replacement string. Specify NULL for a null string.

Default:_

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-reserved_words` *`string`*

Specifies words to be avoided, such as "begin end".

`-restricted` *`string`*

Specifies the list of strings that cannot be used in a name. Separate independent patterns to be replaced with a “space.” These strings are replaced with the `-replace_str` string.

`-skip_bus_net`

Skips nets that are part of busses are changed. Only scalar nets can be changed.

`-subdesign`

Changes the names of subdesign (object).

`-subport_bus`

Changes the names of subport bus objects.

Note: You cannot change the left bracket, “[“, and the right bracket, “]”, because they are a part of the bus name when referencing individual bits of the bus.

`-suffix`

Adds a suffix to the names of the objects to be changed.

`-system_verilog`

Replaces SystemVerilog reserved words.

`-uppertolower`

Changes the names of all objects from uppercase to lowercase. This applies to objects of type instance, port, net, subport, design, and subdesign.

`-verilog`

Replaces Verilog reserved words.

`-vhdl`

Replaces VHDL reserved words as well as strings that start with a digit, an underscore, continuous (two or more) underscores, and end with an underscore. You do not need to specify the `-case_insensitive`, `-instance`, or `-subdesign` option if you specify the `-vhdl` option.

Examples

- The following example adds a suffix `_t` to the design name:

  ```
  legacy_genus:/>change_names -design -suffix _t
  ```

  If the module name of the design was `SAMPLE`, it will be renamed to `SAMPLE_t`.

- The following example replaces all lowercase “`n`” with uppercase “`N`”, and all underscores “_” with hyphens “-” in all instance names.

  ```
  legacy_genus:/> change_names -instance -map {{"n" "N"} {"_" "-"}}
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example replaces all lowercase “a” with uppercase “A” on all subdesigns and subports.

```
legacy_genus:/> change_names -map {{"a" "A"}} -subdesign -subport_bus
```

- The following example replaces any instances of ab, bc, or ca with “@” in all object names.

```
legacy_genus:/> change_names -restricted "ab bc ca" -replace_str "@"
```

- The following example specifies the maximum length of all subdesign names to be 12 characters. Issue this command after elaboration or before writing out the netlist.

```
legacy_genus:/> change_names -max_length 12 -subdesign
```

- The following example ignores case sensitivity. Because the design contains nets n_73 and N_73, Genus renames one of the nets to avoid a naming conflict.

```
legacy_genus:/> mv n_73 N_73
/designs/alu/nets/N_73
legacy_genus:/> mv n_72 n_73
/designs/alu/nets/n_73
legacy_genus:/> change_names -case_insensitive -net
Info    : Change names is successful [CHNM-102]
         : /designs/alu/nets/n_73 moved to /designs/alu/nets/n_73_1
```

- The following example illustrates that you cannot change the brackets (“[“ and “]”) when they are a part of the bus name referencing individual bits of the bus:

```
legacy_genus:/designs/test/ports_in> ls
./       SI2      clk1     in1[0]   in2[0]   in2[3]   in3[2]   in3[5]
legacy_genus:/designs/test/ports_in> change_names -port_bus -map {{"[" "("}
{"]" \
")"}}
legacy_genus:/designs/test/ports_in> ls
./       SI2      clk1     in1[0]   in2[0]   in2[3]   in3[2]   in3[5]
```

- The following two commands both change the brackets (“[” and “]”) in the instance name to “x”s:

```
change_names -instance -restricted {[ ]} -replace_str "x"
change_names -instance -restricted "\[ \]" -replace_str "x"
```

Note: There is no need to escape special characters when enclosing them in braces ({}). If you added the escape character inside the braces, the tool would try to replace this character as well with "x".

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example allows all capital and lower case letters, numbers, underscores, backslashes, and brackets:

```
legacy_genus:/> change_names -allowed \
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_\[\]
```

  Note: You cannot use the dash "-" to indicate inclusion. That is, the following example is not allowed:

```
legacy_genus:/> change_names -allowed a-zA-Z0-9_[]
```

- The following example creates a separate `change_names` log file, `test.log`, that reflects the subdesign name change:

```
legacy_genus:/> change_names -map {{"SUB" "HERO_SUB"}} -subdesign -log_changes
test.log
```

- The following example indicates that the `d` cannot be used as the first character in a net name and that when a name collision would occur, the prefix `xx` can be used.

```
change_names -name_collision -first_res d -prefix xx
```

- The following example shows a part of a netlist that includes VHDL reserved words:

```
generate u1(.A (eg1[0] ), .B (B[0]), .Y (Y[0]));
open u2(.A (eg1[1] ), .B (B[1]), .Y (Y[1]));
endmodule  u3(.A (eg1[2] ), .B (B[2]), .Y (Y[2]));
```

  To change, or eliminate, the names of the VHDL reserved words, use the `-vhdl` option.

```
legacy_genus:/> change_names -vhdl
```

  Now the netlist does not contain any VHDL keywords:

```
generate_cn u1(.A (eg1[0] ), .B (B[0]), .Y (Y[0]));
open_cn u2(.A (eg1[1] ), .B (B[1]), .Y (Y[1]));
endmodule  u3(.A (eg1[2] ), .B (B[2]), .Y (Y[2]));
```

- The following example changes all the object names from lowercase to uppercase:

```
legacy_genus:/> ls /designs/test/ports_in
./    in1[0]    in1[1]    in1[2]    in1[3]    in2[0]    in2[1]    in2[2]
legacy_genus:/> change_names -lowertoupper
Setting in1[3] --> IN1[3
Setting in1[2] --> IN1[2]
Setting in1[1] --> IN1[1]
...
legacy_genus:/> ls /designs/test/ports_in
./    IN1[0]    IN1[1]    IN1[2]    IN1[3]    IN2[0]    IN2[1]
```

  Notice how the lowercase design name change to uppercase as well. The `-uppertolower` option works similarly, except that it changes all uppercase letters to lowercase.

- The following example specifies the character ranges that are allowed when renaming instances.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
change_names -regexp -allowed "a-zA-Z0-9" -instance
```

- The following example specifies that inside hierarchical instance `m2`, any changed names cannot end with the character 2.

```
change_names -last_restricted 2 /designs/m1/instances_hier/m2
```

- The following example uses the `MY_UNCONN_` prefix for dummy nets when writing out the netlist.

```
write_hdl
change_names -dummy_net_prefix MY_UNCONN_
write_hdl
```

Netlist before net name changes:

```
wire [1:0] in1, in2, in3;
wire [5:0] out1, out2;
wire UNCONNECTED, n_0, t;
assign out2[3] = 1'b0;
assign out2[4] = 1'b0;
assign out2[5] = 1'b0;
assign out1[1] = 1'b0;
a a_1(.in1 ({in1[1], 1'b0}), .in2 ({in2[1], 1'b0}), .out1
({out1[5:2], UNCONNECTED, out1[0]}));
```

Netlist after the net name changes:

```
wire \MY_UNCONN_, n_0, t;
assign out2[3] = 1'b0;
assign out2[4] = 1'b0;
assign out2[5] = 1'b0;
assign out1[1] = 1'b0;
a a_1(.in1 ({in1[1], 1'b0}), .in2 ({in2[1], 1'b0}), .out1
       ({out1[5:2], \MY_UNCONN_, out1[0]}));
```

Related Information

Related command: write_hdl on page 404

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_gating

Refer to clock_gating in Chapter 14, “Low Power Synthesis.”

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

connect

```
connect [-net_name string]
     {constant|subport|port|pin|pgpin}
     [-prefix string] [-lenient]
     [-in_prefix string] [-out_prefix string]
     [-remove_multi_driver]
```

Connects two objects in different levels of the hierarchy.

You should specify the driving pin before the load pin.

Options and Arguments

`{constant|subport|port|pin||pgpin}`

Specifies the object to connect.

Only one of the objects can be a `pgpin` object.

`-in_prefix string` Specifies the prefix for new input ports.

`-lenient` Allows the tool to continue even if some pins cannot be connected.

`-net_name string` Specifies the name for the net connection.

`-out_prefix string`

Specifies the prefix for new output ports.

`-prefix string` Specifies the prefix for new ports and nets.

`-remove_multi_driver`

Specifies not to allow multi driver connections.

Related Information

Related commands: edit_netlist connect on page 1324

disconnect on page 1319

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_design

`create_design -name` *`string`* `[-quiet]`

Creates a new design at the same level as the existing top-level design.

The new design is created in the `designs` directory. Once the design is created, you can specify instances, `port_bus`, and so on.

Options and Arguments

`-name` *`string`*	Specifies the name of the new design.
`-quiet`	Suppresses the warning messages regarding naming conflicts.

Related Information

Related command:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_hport_bus

```
create_hport_bus -name string
     [-left_bit integer] [-right_bit integer]
     {-input|-output}
     {instance}
```

Creates a `hport_bus` in a design. The command can also create a single subport by omitting both the `-left_bit` and `-right_bit` options.

Options and Arguments

`-input`	Creates an input `hport_bus`.
`{instance}`	Specifies the name of the instance for which to create the `hport_bus`.
`-left_bit integer`	Specifies the leftmost bit index of the `hport_bus`.
`-name string`	Specifies the name of the new `hport_bus`.
`-output`	Creates an output `hport_bus`.
`-right_bit integer`	Specifies the rightmost bit index of the `hport_bus`.

Related Information

Related command: edit_netlist new_instance on page 1336

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_inst

```
create_inst [-name string]
     {design|subdesign|lib_cell|base_cell}
     {subdesign|design} [-quiet]
```

Creates a specified instance type in a specified level of design hierarchy. You can instantiate inside a top-level design or a subdesign.

- You cannot instantiate objects that require a change to a preserved module.
- You cannot create a hierarchical loop.

  If subdesign `A` contains subdesign `B`, you cannot instantiate `A` underneath `B`.

Options and Arguments

-name *string*
: Specifies the name of the new instance.

{*design*|*subdesign*|*lib_cell*|*base_cell*}
: Specifies the object to instantiate.

-quiet
: Suppresses the warning messages regarding naming conflicts.

{*subdesign*|*design*}
: Specifies the name of the design or subdesign in which you want to instantiate the object.

Command Information

While this command is inserting a new instance, it operates only on designs, subdesigns, or libcells. If inserting a design or subdesign, the design/subdesign must already exist. This can be created by reading in a new design and elaborating, reading in a netlist, or by using `create_design` to create it.

If inserting a new libcell, make sure to use a libcell from the same library domain. The libcell name must be a full Genus database virtual directory name.

The new instance is inserted with no connections - all inputs and outputs are floating. After using this command, use connect to make the appropriate connections.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_port_bus

```
create_port_bus -name string
     [-left_bit integer] [-right_bit integer]
     {-input|-output}
     [design]
```

Creates a `port_bus` object in a design. The command can also create a single port by omitting both the `-left_bit` and `-right_bit` options.

Options and Arguments

`design`	Specifies the name of the design for which to create the `port_bus`. The design name can be omitted if there is only one top-level design.
`-input`	Creates an input `port_bus`.
`-input -output`	Creates a bidirectional `port_bus`.
`-left_bit` *`integer`*	Specifies the leftmost bit index of the bus.
`-name` *`string`*	Specifies the name of the new `port_bus`.
`-output`	Creates an output `port_bus`.
`-right_bit` *`integer`*	Specifies the rightmost bit index of the bus.

Related Information

Related command: create_design on page 1310

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_primitive

```
create_primitive [-name string] [-inputs integer]
     [-quiet] logic_function {design|subdesign}
```

Creates an unmapped primitive cell in a design or a subdesign.

Options and Arguments

`{design|subdesign}`

Specifies the name of the design or subdesign in which you want to instantiate the primitive cell. If not specified, the tool will pick a name.

`-inputs integer`

Specifies the number of input pins to create for the primitive cell. This option is only valid for logic functions and, nand, or, nor, xor, xnor, mux, and bmux.

`logic_function`

Specifies the logic function of the primitive cell. You can specify any of the following:

and	latch	notif1	bmux
buf	nand	or	dc
bufif0	nor	xnor	
bufif1	not	xor	
d_flop	notif0	mux	

`-name string`

Specifies the name of the new primitive cell.

`-quiet`

Suppresses the warning messages regarding naming conflicts.

Note: If you specify same name for two primitive cells created under the same hierarchy, then Genus will automatically add a number to the suffix of the second primitive cell's name along with a warning regarding naming conflicts, which can be suppressed by using this option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Primitive Functions and Pins

- **and, nand, nor, or, xor, and xnor** - These will have input widths based on the value provided with `-inputs`. The input pins start as `in_0`, and go up to the `in_`*`value-1`*. The output pin is `z`.
- **buf and not** - These will have one input pin: `in_0`. And it will have one output pin: `z`
- **d_flop** - This instantiates a generic non-scan flop. This flop has input pin functions to define the behavior. These pins will all be under `pins_in` on the newly created instance
  - **clk** - clock pin for the flop.
  - **d** - data input pin.
  - **sena** - enable pin. When this signal is high, d will be clocked through. When low, q is maintained. Tie to 1 if not using.
  - **aclr** - asynchronous clear. When high, q will immediately go to 1'b0. Tie to 0 if not using.
  - **apre** - asynchronous set. When high, q will immediately go to 1'b1, Tie to 0 if not using.
  - **srl** - synchronous reset. When high, on a clock edge, q will take the value of srd. Tie to 0 if not using.
  - **srd** - synchronous reset data value. Tie to 1 or 0 depending on your requirements for synchronous reset. Tie to 0 if not using.
    This output pin will be under `pins_out` on the newly created instances.
  - **q** - q data output
- **latch** - This instantiates a generic latch. This latch has the standard latch pins that will be found under `pins_in` on the newly created instance
  - **ena** - active high enable pin for the latch
  - **d** - data input pin.
  - **aclr** - asynchronous clear. When high, q will immediately go to 1'b0. Tie to 0 if not using.
  - **apre** - asynchronous set. When high, q will immediately go to 1'b1, Tie to 0 if not using.
    This output pin will be under `pins_out` on the newly created instances.
  - **q** - q data output

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- **mux** - This instantiates a standard mux. Each bit has an individual control signal. As a result, the minimum -input value is 4 and the value must be even. The inputs pins will be found under `pins_in` on the new instance. The selects are active high. If no selects are active, the output will be 0 or X (if the mux is defined to be one hot).
  - **sel*n*** - active high select pin. The value *n* starts at 0 and goes to up to the value of `-input` divided by 2
  - **data*n*** - data pin. The value *n* starts at 0.
    This output pin will be under `pins_out` on the newly created instances.
  - **z** - data output
- **bmux** - This instantiates a binary mux with encoded selects. The tool will determine the mix of selects and inputs based on the value passed to `-input`. For example, if the value is 10, there will be 3 selects and 7 data inputs. If the value is 11, there will still be 3 selects, but now 8 data inputs. The inputs pins will be found under `pins_in` on the new instance. The selects are active high.
  - **sel*n*** - active high select pin. The value *n* starts at 0.
  - **data*n*** - data pin. The value *n* starts at 0
    This output pin will be under `pins_out` on the newly created instances.
  - **z** - data output
- **dc** - This instantiates a don't care generator that can be used for logic optimization. Use of this primitive is rare. The input pins can be found under `pins_in` on the new instance
  - **cf** - data value to be passed through when dcf is 0
  - **dcf** - control pin. When low, dcf is passed to z. When high, 1'bx is passed to z
    This output pin will be under `pins_out` on the newly created instances.
  - **z** - data output

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

dedicate_module

`dedicate_module` *`instance`* `[`*`instance`*`]...`

Creates a new subdesign by copying the subdesign that is common to the listed hierarchical instances. The command returns the path to the newly created subdesign.

This command is applicable for hierarchical instances but not for leaf level instances.

The creation of a new subdesign allows you to make changes that affect a limited set of instances instead of all instances of the original subdesign.

Options and Arguments

`instance`	Specifies the name of a hierarchical instance for which you want a dedicated subdesign. You must specify a list of instances that share the same subdesign. This option accepts Tcl lists and wildcards (*,?).

Related Information

Related commands:	edit_netlist uniquify on page 1343 uniquify on page 1359

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

delete_unloaded_undriven

```
delete_unloaded_undriven
     [-disconnect] [-force_bit_blast] [-all] [-inout]
     [design] [> file]
```

Disconnects subports and hierarchical pins connected to constants and that do not fanout to anything, and deletes unloaded and undriven subports from the design. Use this command as a post-processing step to remove any unused subports from the netlist.

By default the command skips individual bits of a bus that are connected to constants or that are unused.

Options and Arguments

`-all`	Additionally, deletes unloaded and undriven top-level ports from the design.
`design`	Specifies the design from which to remove the unused ports or subports.
`-disconnect`	Only disconnects the subports and hierarchical pins that are connected to constants and that do not fanout to anything.
`file`	Specifies the name of the file to which the output of the command should be redirected.
`-force_bit_blast`	Bitblasts modules that have individual bus bits that are unused or connected to constants and deletes the unused bus bits.
`-inout`	Specifies to delete disconnected inout pins too.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

disconnect

`disconnect {pin|pgpin|port|subport}...`

Disconnects multiple pins, ports, or subports from each object they are connected to. For example, if `A`, `B`, and `C` are connected together and you disconnect `A`, then `B` and `C` remain connected to each other, but `A` is now left unconnected.

You cannot disconnect an object that would require changes to a preserved module.

You cannot disconnect a generic constant (1 or 0) pin of the module but you can disconnect the loads from that pin.

You can disconnect an object that is not currently connected to anything else. In that case nothing happens.

Options and Arguments

pin	Specifies the name of an instance pin to disconnect.
pgpin	Specifies the name of the power or ground pin to disconnect.
port	Specifies the name of a design port to disconnect.
subport	Specifies the name of a subport (port on a hierarchical instance) to disconnect.

Related Information

Related commands:

edit_netlist connect on page 1324

edit_netlist disconnect on page 1329

edit_netlist hier_connect on page 1334

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist

```
edit_netlist { bitblast_all_ports | connect
     | dedicate_subdesign | disconnect
     | group | new_design | new_instance
     | new_port_bus | new_primitive | new_subport_bus
     | ungroup | uniquify}
```

Edits a gate-level design.

Options and Arguments

`bitblast_all_ports`	Bitblasts all ports of a design or subdesign.
`bitblast_port`	Bitblasts a port_bus (or subport_bus) of a design or hierarchical instance.
`connect`	Connects a pin, port or subport to another pin, port or subport.
`dedicate_subdesign`	Replaces a subdesign of instances with a dedicated copy.
`delete`	Removes an object from the design hierarchy.
`disconnect`	Disconnects a pin, port or subport.
`group`	Builds a level of hierarchy around instances.
`hier_connect`	Connects two objects in different levels of the hierarchy.
`new_design`	Creates a new design.
`new_instance`	Creates a new instance.
`new_port_bus`	Creates a new `port_bus` on a design.
`new_primitive`	Creates a new unmapped primitive instance.
`new_subport_bus`	Creates a new `subport_bus` on a hierarchical instance.
`ungroup`	Flattens a level of hierarchy
`uniquify`	Eliminates sharing of subdesigns between instances

Related Information

Related commands: edit_netlist bitblast_all_ports on page 1322

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist bitblast_port on page 1323
edit_netlist connect on page 1324
edit_netlist dedicate_subdesign on page 1326
edit_netlist delete on page 1328
edit_netlist disconnect on page 1329
edit_netlist group on page 1332
edit_netlist hier_connect on page 1334
edit_netlist new_design on page 1335
edit_netlist new_instance on page 1336
edit_netlist new_port_bus on page 1338
edit_netlist new_primitive on page 1339
edit_netlist new_subport_bus on page 1341
edit_netlist uniquify on page 1343
edit_netlist group on page 1332
edit_netlist ungroup on page 1342

Affected by this attribute: ui_respects_preserve

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist bitblast_all_ports

`edit_netlist bitblast_all_ports {design|subdesign}...`

Bitblasts all ports of the specified design or subdesign. This command is available after elaboration. The name of the bitblasted ports will follow the nomenclature specified by the `bit_blasted_port_style` attribute. The default bit blasted port style is:

```
%s_%d
```

Options and Arguments

`{design|subdesign}`	Specifies the design or subdesign in which the ports should be bitblasted.

Example

In the following example, the Verilog design `top` has a four-bit input port named `AI`:

```
AI[0:3]
```

The `edit_netlist bitblast_all_ports` command is issued on the design `top`, bitblasting the `AI` port:

```
...
legacy_genus:/> elaborate
...
legacy_genus:/> edit_netlist bitblast_all_ports top
legacy_genus:/> ls /designs/top/ports_in

AI_0 AI_1 AI_2 AI_3
```

Related Information

Affected by this attribute:	bit_blasted_port_style
Related Attributes:	write_vlog_bit_blast_bus_connections
	write_vlog_bit_blast_mapped_ports

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist bitblast_port

```
edit_netlist bitblast_port
     {port_bus | subport_bus}
     {design | instance}
```

Bitblasts the specified port_bus (or subport_bus) of the specified design (or hierarchical instance). This command is available after elaboration. The name of the bitblasted ports will follow the nomenclature specified by the `bit_blasted_port_style` attribute. The default style is:

```
%s_%d
```

Options and Arguments

design	Specifies the design to which the port_bus belongs.
instance	Specifies the hierarchical instance to which the subport_bus belongs.
{*port_bus*|*subport_bus*}	Specifies the name of the port_bus or subport_bus to be bitblasted.

Related Information

Affected by this attribute:	bit_blasted_port_style
Related Attributes:	write_vlog_bit_blast_bus_connections
	write_vlog_bit_blast_mapped_ports

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist connect

```
edit_netlist connect
     {constant|pin|pgpin|port|subport}
     {constant|pin|pgpin|port|subport}
     [-remove_multi_driver]
     [-net_name string] [-lenient]
```

Connects the two specified objects, and anything to which they might already be connected.

You can create nets that have multiple drivers, and you can create combinational loops.

The `logic0` and `logic1` pins are visible in the directory so that you can connect to them and disconnect from them. They are in a directory called `constants` and are called `1` and `0`. The following example shows how the top-level `logic1` pin appears in a design called `add`:

```
/designs/add/constants/1
```

The following example shows how a `logic0` pin appears deeper in the hierarchy:

```
/designs/add/instances_hier/bad/constants/0
```

Each level of hierarchy has its own dedicated logic constants that can only be connected to other objects within that level of hierarchy.

The command has a number of limitations. Violation of the following limitations will generate error messages and cause the command to fail. You cannot connect

- Pins, ports, or subports that are in different levels of hierarchy.
- Pins, ports, or subports that are already connected
- An object to itself.
- An object that is driven by a logic constant to an object that already has a driver. This prevents you from shorting the logic constant nets together.
- Objects if it would require a change to a preserved module.

Options and Arguments

constant	Specifies the name of the constant to connect.
`-lenient`	Allows the tool to continue even if some pins cannot be connected.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-net_name` *`string`*	Specifies the user-defined name of the net.
`pin`	Specifies the name of an instance pin to connect.
`pgpin`	Specifies the name of the power or ground pin to connect.
`port`	Specifies the name of a design port to connect.
`-remove_multi_driver`	Specifies not to allow multi driver connections.
`subport`	Specifies the name of a subport (port on a hiearchical instance) to connect.

Example

- In the following example, `A` and `B` are already connected and `C` and `D` are already connected. When you connect `A` and `C`, the result is a net connecting `A`, `B`, `C`, and `D`.

```
legacy_genus:/designs/alu/ports_in> edit_netlist connect A C
/designs/alu/nets/A_
```

Related Information

Related command: edit_netlist disconnect on page 1329

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist dedicate_subdesign

`edit_netlist dedicate_subdesign` *`instance`* `[`*`instance`*`]...`

Creates a dedicated subdesign for a particular hierarchical instance, when a subdesign is instantiated multiple times in the design. Effectively, this command creates a copy of the specified instance's subdesign, and reassigns the instance to the new subdesign.

This command is applicable for hierarchical instances but not for leaf instances.

Note: The `uniquify` command creates dedicated sub-module for all the instances, where as `dedicate_module` creates a separate module definition only for the instances mentioned with the command.

Options and Arguments

`instance`	Specifies the name of a hierarchical instance for which you want a dedicated subdesign. If a list of instances is specified, all the instances in the list will get the same new subdesign You must specify a list of instances that share the same subdesign. This option accepts TCL lists and wildcards (*, ?).

Example

- This example shows the basic use of dedicated subdesign:

  In the following example the design top contains a module *sub* that has been instantiated five times in the design. The instance names are `sub1`,`sub2`, `sub3`, `sub4`, and `sub5`. To create a separate subdesign for instances `sub1` and `sub2`, enter the following command:

  ```
  legacy_genus:/> edit_netlist dedicate_subdesign {/designs/top/instances_hier/
  sub1 \
      /designs/top/instances_hier/sub2}
  ```

  Note: If you would execute the `edit_netlist dedicate_subdesign` command on the remaining three instances, no new subdesign would be created because they already share a subdesign that is not used by any other instances.

- This example shows creating a new subdesign that is used by two instances:

  ```
  legacy_genus:/designs/dtmf_rcvr_core> edit_netlist dedicate_subdesign {\
  /designs/dtmf_rcvr_core/instances_hier/g4157 /designs/dtmf_rcvr_core/ \
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
instances_hier/g4158}
/designs/dtmf_rcvr_core/subdesigns/my_buf_5
```

#See that the instances are assigned correctly

```
legacy_genus:/designs/dtmf_rcvr_core> get_attr subdesign /designs/
dtmf_rcvr_core/instances_hier/g4157
50

/designs/dtmf_rcvr_core/subdesigns/my_buf_5
51

legacy_genus:/designs/dtmf_rcvr_core> get_attr subdesign /designs/
dtmf_rcvr_core/instances_hier/g4158
52

/designs/dtmf_rcvr_core/subdesigns/my_buf_5
```

Related Information

Related commands: edit_netlist uniquify on page 1343

uniquify on page 1359

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist delete

`rm object... [-quiet]`

Removes an object from the design hierarchy. This command is similar to its UNIX counterpart.

For a current list of the objects that can be removed, refer to the command help.

If the hierarchical pin or port bus object has a net connection, the net is disconnected first and then the object is removed.

If you remove a design, the CPF-related information will also be removed from the design hierarchy.

Note: This command does not work on the `pin` or `port` object.

Alias for rm.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist disconnect

`edit_netlist disconnect {pin|pgpin|port|subport}...`

Disconnects multiple pins, ports, or subports from each object they are connected to. For example, if `A`, `B`, and `C` are connected together and you disconnect `A`, then `B` and `C` remain connected to each other, but `A` is now left unconnected.

You cannot disconnect an object that would require changes to a preserved module.

You cannot disconnect a generic constant (1 or 0) pin of the module but you can disconnect the loads from that pin.

You can disconnect an object that is not currently connected to anything else. In that case nothing happens.

Options and Arguments

pin	Specifies the name of an instance pin to disconnect.
pgpin	Specifies the name of the power or ground pin to disconnect.
port	Specifies the name of a design port to disconnect.
subport	Specifies the name of a subport (port on a hiearchical instance) to disconnect.

Examples

- The following example disconnects input port `data[4]`.

```
legacy_genus:/designs/alu/ports_in> edit_netlist disconnect data[4]
```

- The following example shows how you can disconnect a constant pin 1 from all its loads.

```
set cnet [get_attr net /designs/test/constants/1]
set all_loads [get_attr loads $cnet]
foreach load $all_loads {
    edit_netlist disconnect $load
}
```

  Note: If the constant pin has a large number of loads, disconnecting each of these loads may impact runtime.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: edit_netlist connect on page 1324

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist flatten_complex_ports

```
edit_netlist flatten_complex_ports
     [module | design] ...
```

Converts two-dimensional port arrays into one-dimensional port arrays.

Options and Arguments

module | *design*	Specifies the module(s) or design for which the port conversion needs to be done. By default, all ports of all modules including top module get converted.

Example

The following command converts the two-dimensional array `my_port[7:0][3:0]` to one-dimensional array `my_port[31:0]` if `my_port` is a port of module A.

```
edit_netlist flatten_complex_ports A
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist group

```
edit_netlist group -group_name group_name
     instance [instance]...
```

Creates a level of the design hierarchy by grouping the specified instances. You can only group instances that belong to the same hierarchy.

Options and Arguments

`-group_name` *`group_name`*

Specifies the name of the module that groups the specified instances.

By default, the resulting module will have an instance name consisting of the specified group name with the suffix `i`, and is placed in the `instances_hier` directory. The suffix is used to indicate that this hierarchy is the result of a group command.

You can change the suffix with the `group_instance_suffix` attribute.

`instance`

Specifies the name of an instance to add to the specified group. You need to specify at least one instance.

Examples

- The following example groups instances `accum_1` and `averg_1` into one module `my_module`.

```
legacy_genus:/> edit_netlist group -group_name my_module accum_1 averg_1
/designs/alu/instances_hier/my_modulei
```

- The following command returns an error because the specified instances do not belong to the same hierarchy.

```
legacy_genus:/> edit_netlist group [find / -instance m5] [find / -instance
m3_0]
Error   : Not all instances belong to the same hierarchy. [TUI-234] [edit_netlist
group]
        : Instance '/designs/m1/instances_hier/m2/instances_hier/m3/
instances_hier/m4/instances_hier/m5' is part of (sub)design 'm4'. Instance '/
designs/m1/instances_hier/m2/instances_hier/m3/instances_hier/m3_0' is not part of
(sub)design 'm4'.
        : The 'edit_netlist group' command can only group instances contained
within the same hierarchy.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command:	edit_netlist ungroup on page 1342
Affected by these attributes:	group_generate_portname_from_netname
	group_instance_suffix

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist hier_connect

```
edit_netlist hier_connect
     {constant|subport|port|pin|pgpin}
     {constant|subport|port|pin|pgpin}
     [-prefix string ]
     [-in_prefix string] [out_prefix string]
```

Connects two objects in different levels of the hierarchy.

Note: You should specify the driving pin before the load pin.

Options and Arguments

`{constant|subport|port|pin|pgpin}`
: Specifies the object to connect.

`-in_prefix string`
: Specifies the prefix for new input ports.

`-out_prefix string`
: Specifies the prefix for new ouput ports.

`-prefix string`
: Specifies the prefix for ports and nets.

Example

The following command connects the input pins of two inverters in the hierarchical instances aa1 and aa2.

```
edit_netlist hier_connect [find . -pin aa1/inv/in_0] [find . -pin aa2/inv/in_0]
```

Related Information

Related command: edit_netlist connect on page 1324

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist new_design

`edit_netlist new_design -name string [-quiet]`

Creates a new design at the same level as the existing top-level design.

The new design is created in the `/designs` directory. Once the design is created, you can specify instances, `port_bus`, and so on.

Options and Arguments

`-name string`	Specifies the name of the new design.
`-quiet`	Suppresses the warning messages regarding naming conflicts.

Examples

- The following example creates a new design called `DESIGNA`.

  ```
  legacy_genus:/> edit_netlist new_design -name DESIGNA
  ```

  The new design `DESIGNA`, will be created in the `/designs` directory. Once the design is created, the instances, `port_bus`, and so on can be specified.

- The following example tries to create a new design called `TEST`. However, a design by that name already exists. In such cases, a number will be appended to the end of the specified name and an error message indicating the naming conflict will be printed. The following example specifies the `-quiet` option to suppress this warning.

  ```
  legacy_genus:/> edit_netlist new_design -name TEST -quiet
  /designs/TEST1
  ```

  The name given to the new design in this case is `TEST1`.

Related Information

Related command: edit_netlist new_port_bus on page 1338

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist new_instance

```
edit_netlist new_instance [-name string]
     {design|subdesign|libcell}
     {subdesign|design} [-quiet]
```

Creates a specified instance type in a specified level of design hierarchy. You can instantiate inside a top-level design or a subdesign.

- You cannot instantiate objects that require a change to a preserved module.
- You cannot create a hierarchical loop.

  If subdesign `A` contains subdesign `B`, you cannot instantiate `A` underneath `B`.

Options and Arguments

`-name` *`string`*	Specifies the name of the new instance.
`{`*`design`*`|`*`subdesign`*`|`*`libcell`*`}`	Specifies the object to instantiate.
`-quiet`	Suppresses the warning messages regarding naming conflicts.
`{`*`subdesign`*`|`*`design`*`}`	Specifies the name of the design or subdesign in which you want to instantiate the object.

Examples

- The following example creates a new instance called `TEST_SUB` under the `TEST` design:

```
legacy_genus:/> edit_netlist new_instance -name TEST_SUB /designs/TEST \
    /designs/TEST
legacy_genus:/> ls /designs/TEST/instances_hier/
/designs/TEST/instances_hier/TEST_SUB
```

- The following example tries to create a new subdesign called `TEST_SUB` under the `TEST` design. However, a subdesign by that name already exists. In such cases, a number will be appended to the end of the specified name and an error message indicating the naming conflict will be printed. The following example specifies the `-quiet` option to suppress this warning.

```
legacy_genus:/> edit_netlist new_instance -name TEST_SUB /designs/TEST -quiet
\
        /designs/TEST
/designs/TEST/instances_hier/TEST_SUB3
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The name given to the new subdesign in this case is `TEST_SUB3`.

Related Information

Related command: edit_netlist new_subport_bus on page 1341

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist new_port_bus

```
edit_netlist new_port_bus -name string
     [-left_bit integer] [-right_bit integer]
     {-input|-output|-input -output}
     [design]
```

Creates a `port_bus` object in a design. The command can also create a single port by omitting both the `-left_bit` and `-right_bit` options.

Options and Arguments

design	Specifies the name of the design for which to create the port_bus. The design name can be omitted if there is only one top-level design.
`-input`	Creates an input port_bus.
`-input -output`	Creates a bidirectional port_bus.
`-left_bit` *integer*	Specifies the leftmost bit index of the bus.
`-name` *string*	Specifies the name of the new port_bus.
`-output`	Creates an output port_bus.
`-right_bit` *integer*	Specifies the rightmost bit index of the bus.

Example

- The following example creates a single input port named `a_in`:

```
legacy_genus:/> edit_netlist new_port_bus -name a_in -input
```

Related Information

Related command: edit_netlist new_design on page 1335

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist new_primitive

```
edit_netlist new_primitive [-name string] [-inputs integer]
     [-quiet] logic_function {design|subdesign}
```

Creates an unmapped primitive cell in a design or a subdesign.

Options and Arguments

`{design|subdesign}`

Specifies the name of the design or subdesign in which you want to instantiate the primitive cell.

`-inputs integer`

Specifies the number of input pins to create for the primitive cell.

`logic_function`

Specifies the logic function of the primitive cell. You can specify any of the following:

`and`	`latch`	`notif1`
`buf`	`nand`	`or`
`bufif0`	`nor`	`xnor`
`bufif1`	`not`	`xor`
`d_flop`	`notif0`	

`-name string`

Specifies the name of the new primitive cell.

Note: If you specify same name for two primitive cells created under the same hierarchy, then Genus will automatically add a number to the suffix of the second primitive cell name along with a warning regarding naming conflicts, which can be suppressed by using the `-quiet` option.

`-quiet`

Suppresses the warning messages regarding naming conflicts.

Examples

- The following example creates a new buffer instance called `I101` in design `DESIGNA`:

```
legacy_genus:/> edit_netlist new_primitive -name I101 buf DESIGNA
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The new instance, `I101`, is created in the `/designs/DESIGNA/instances_comb` directory.

A sequential primitive (`d_flop` or `latch`) will be created in the `instances_seq` directory.

- The following example tries to create a new buffer instance called `I101`. However, a buffer by that name already exists. In such cases, a similar name will be chosen and an error message indicating the naming conflict will be printed. The following example specifies the `-quiet` option to suppress this warning:

  ```
  legacy_genus:/> edit_netlist new_primitive -name I101 buff TEST -quiet
  /designs/TEST/instances_comb/I1
  ```

  The name given to the new buffer in this case is `I1`.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist new_subport_bus

```
edit_netlist new_subport_bus -name string
     [-left_bit integer] [-right_bit integer]
     {-input|-output|-input -output}
     instance
```

Creates a `subport_bus` in a design. The command can also create a single subport by omitting both the `-left_bit` and `-right_bit` options.

Options and Arguments

design	Specifies the name of the instance for which to create the `subport_bus`.
`-input`	Creates an input `subport_bus`.
`-input -output`	Creates a bidirectional `subport_bus`.
`-left_bit` *integer*	Specifies the leftmost bit index of the `subport_bus`.
`-name` *string*	Specifies the name of the new `subport_bus`.
`-output`	Creates an output `subport_bus`.
`-right_bit` *integer*	Specifies the rightmost bit index of the `subport_bus`.

Example

- The following example creates a single input subport named `a_in`:

```
legacy_genus:/> edit_netlist new_subport_bus -name a_in -input
```

Related Information

Related command: edit_netlist new_instance on page 1336

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist ungroup

```
edit_netlist ungroup [-prefix string] instance...
```

Removes a level of the design hierarchy.

Large numbers of small hierarchical blocks in a design can sometimes limit optimization since the hierarchical boundaries must be preserved. Many hierarchical blocks may also increase the memory usage since information about each block and port names must be stored. The `ungroup` command provides a mechanism to remove these unwanted levels of hierarchy.

Options and Arguments

instance	Specifies the hierarchical instance for which to remove one level of the hierarchy. The components of the specified instance then become instances in the parent block.
`-prefix`	Specifies a prefix for the ungrouped instances.

Examples

- The following example ungroups all hierarchical instances whose names end in `_little`:

```
legacy_genus:/> edit_netlist ungroup [find / -instance *_little]
```

- The following example ungroups the instance `inst1` and specifies that the resulting ungrouped instances of `inst1` have a prefix of `inst1_test`. Using the prefix allows you to identify from which instance the ungrouped instances originated:

```
legacy_genus:/designs/test/instances_hier> edit_netlist ungroup -prefix
inst1_test \
    inst1
legacy_genus:/designs/test/instances_comb> ls


inst1_test_g1/  inst1_test_g2/  inst1_test_g3/
```

Related Informations

Related command: edit_netlist group on page 1332

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

edit_netlist uniquify

```
edit_netlist uniquify
     {subdesign|design} [-verbose]
```

Uniquifies the instances under the specified design or subdesign. Uniquification is the process of creating a new subdesign for an instance or a group of instances. The newly created subdesign is merely a copy of the subdesign to which the original instance or group of instances were associated. That is, an instance or a group of instances will now be a part of their own, *unique* subdesign. The newly created subdesign will usually maintain the original design or subdesign name followed by a number suffix.

Note: Preserved modules cannot be uniquified.

Options and Arguments

{*design*|*subdesign*}	The instances under the specified design or subdesigns will be uniquified.
-verbose	Prints out the uniquified instances and their corresponding subdesign names.

Examples

- The following example has a top level design called `top` with two subdesigns named `A` and `B`:

  ```
  legacy_genus:/designs/top/subdesigns> ls
  ./     A/     B/
  ```

  In order to uniquify the subdesigns, issue the `edit_netlist uniquify` command on `top`:

  ```
  legacy_genus:/designs/top/subdesigns> edit_netlist uniquify /designs/top
  legacy_genus:/designs/top/subdesigns> ls
  ./     A/     A_1/     B/     B_1/
  ```

- The following example shows how to uniquify all subdesigns except for subdesign `mysubdesign`.

  ```
  set_attribute preserve true [find / -subdesign mysubdesign]
  foreach el [find / -subdesign *] {
    edit_netlist uniquify $el
  }
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

flatten_complex_ports

```
flatten_complex_ports
     [subdesign | design] ...
```

Converts two-dimensional port arrays into one-dimensional port arrays.

Options and Arguments

subdesign|*design*	Specifies the module(s) or design for which the port conversion needs to be done. By default, all ports of all modules including top module get converted.

Example

- The following command converts the two-dimensional array `my_port[7:0][3:0]` to one-dimensional array `my_port[31:0]` if `my_port` is a port of module A.

  ```
  flatten_complex_ports A
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

group

```
edit_netlist group -group_name group_name
     instance [instance]...
```

Creates a level of the design hierarchy by grouping the specified instances. You can only group instances that belong to the same hierarchy.

Alias for edit_netlist group.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

insert_tiehilo_cells

```
insert_tiehilo_cells
     [-hilo libcell | -hi libcell | -lo libcell]
     [-aon_hilo libcell | -aon_hi libcell | -aon_lo libcell]
     [-allow_inversion] [-maxfanout integer]
     [-all] [-skip_unused_hier_pins] [-place_cells]
     [-verbose] [subdesign | design]
```

Ties the constants `1'b0` and `1'b1` in the netlist to tie high and tie low cells, respectively. In multiple supply voltage (MSV) designs, this command inserts cells by domain. It skips scan pins, preserved pins, preserved nets, and modules by default. Scan pins can be connected by using the `-all` option.

Use the `ui_respects_preserve` root attribute to override preserve settings.

Tip

Ensure that the specified tie hilo libcell, or tie high and tie low libcells are usable. Make sure that both the `preserve` and `avoid` libcell attributes are set to `false` on the specified cells.

This command should only be run on uniquified designs.

Options and Arguments

`-all`	Inserts tie hi/lo cells without skipping scan pins.
`-allow_inversion`	Allows to use a tie cell with inverter if either the tie high or tie low cell cannot be found in the library.
`-aon_hilo` *`libcell`* | `-aon_hi` *`libcell`* | `-aon_lo` *`libcell`*	Specifies the always-on libcell(s) to replace constant 1s and constant 0s. Either specify one libcell to be used for both constant 1s and constant 0s, or specify a specific libcell to be used for the constant 1s and one to be used for the constant 0s. *Default*: If you specified neither the `-aon_hilo` option, nor the `-aon_hi` and `-aon_lo` options, the tool will use any appropriate always-on cell from the library.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hilo` *`libcell`* | `-hi` *`libcell`* | `-lo` *`libcell`*

Specifies the libcell(s) to replace constant 1s and constant 0s. Either specify one libcell to be used for both constant 1s and constant 0s, or specify a specific libcell to be used for the constant 1s and one to be used for the constant 0s

Default: If you specified neither the `-hilo` option, nor the `-hi` and `-lo` options, the tool will use the first appropriate cell.

`-maxfanout` *`integer`*

Specify the maximum fanout allowed per tie cell.

If this option is not specified, there is no constraint on the fanout.

`-place_cells`

Places the inserted tie cells.

`-skip_unused_hier_pins`

Skips hierarchical constant connected pins which are not used inside the module.

{*`subdesign`* | *`design`*}

Specifies the design or subdesign in which to insert constants.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-verbose`

Provides detailed information of the preserved and scan pins that were skipped in the tie hi/lo cell insertion process.

Examples

- The following example ties the constant `1`s and `0`s to the cells named `TIEHI` and `TIELOW`, respectively. The maximum fanout per tie cell is 10. Using the `-verbose` option shows that two scan pins were skipped:

```
legacy_genus:/> insert_tiehilo_cells -hi TIEHI -lo TIELO  -maxfanout 10
-verbose
```

```
pin: /libraries/slow/libcells/TIELO/Y function: 0
pin: /libraries/slow/libcells/TIEHI/Y function: 1
Connecting all 1'b0 and 1'b1 to TIELO/TIEHI cells.
tielo_cell is /libraries/slow/libcells/TIELO , tiehi_cell is /libraries/slow/
libcells/TIEHI

Info : 2 scan pins which are loads of '0' in /designs/m1 are skipped.
Use the '-all' option to avoid skipping of scan pins.
/designs/m1/instances_seq/foo/bx_reg/pins_in/SE
/designs/m1/instances_seq/tm_reg/pins_in/SE

Done connecting 1'b0 and 1'b1 to TIELO/TIEHI cells
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following example shows two modules, `UI` and `top`. When the `insert_tiehilo_cells -skip_unused_hier_pins` command is used, the pin `B` of the instantiation of `U1` in module `top` will be skipped.

```
module U1(A, B, C, Sel, Z);
    input A, B, C;
    input [1:0] Sel;
    output Z;
    wire A, B, C;
    wire [1:0] Sel;
    wire Z;
    wire n_0, n_1, n_2, n_3, n_4, n_5;
    NAND2X1 g27(.A (n_5), .B (n_4), .Y (Z));
    NAND2X1 g28(.A (n_3), .B (n_2), .Y (n_5));
    NAND2X1 g29(.A (n_1), .B (A), .Y (n_4));
    NOR2X1 g30(.A (n_0), .B (Sel[0]), .Y (n_3));
    INVX1 g31(.A (n_1), .Y (n_2));
    NOR2X1 g32(.A (Sel[1]), .B (Sel[0]), .Y (n_1));
    INVX1 g33(.A (C), .Y (n_0));
endmodule

module top(a, b, c, sel, z);
    input a, b, c;
    input [1:0] sel;
    output z;
    wire a, b, c;
    wire [1:0] sel;
    wire z;
    U1 inst_U1(.A (a), .B (1'b0), .C (c), .Sel (sel), .Z (z));
endmodule
```

Related Information

Related attributes: ui_respects_preserve
use_tiehilo_for_const

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

mv

`mv` *`object new_name`* `[-flexible] [-slash_ok]`

Renames an object in the design hierarchy. This command is similar to its UNIX counterpart.

You can rename the following objects:

- design
- instance
- isolation_rule
- level_shifter_group
- level_shifter_rule
- library_domain
- net
- port_bus
- power_domain
- scan_chain
- scan_segment
- subdesign
- subport_bus
- test_clock
- test_clock_domain
- test_signal

Options and Arguments

`-flexible`	Indicates to be flexible for renaming when there is a collision.
`new_name`	Specifies the new name for the specified object.
`object`	Specifies the object to rename.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-slash_ok`	Indicates that the destination name can have embedded slashes.

Examples

- The following example changes the name of design `comp` to `comp_test`.

```
legacy_genus:/designs> ls
comp
legacy_genus:/designs> mv comp comp_test
legacy_genus:/designs> ls
comp_test
```

- The following example changes the subdesign `mux` to `muxYYY`. You do not have to specify the path name of the target object, just the basename:

```
legacy_genus:/> mv /designs/dpldalgn/subdesigns/mux muxYYY
legacy_genus:/> ls /designs/dpldalgn/subdesigns/
muxYYY
```

- The following example attempts to rename instance `aluout_reg_0` to an already existing instance `aluout_reg_1`. Using the `-flexible` option, the instance gets renamed to `aluout_reg585`, which does not cause a conflict:

```
legacy_genus:/designs/alu/instances_seq> mv aluout_reg_0 aluout_reg_1
-flexible
/designs/alu/instances_seq/aluout_reg585
legacy_genus:/designs/alu/instances_seq> ls
./              aluout_reg_1/   aluout_reg_3/   aluout_reg_5/   aluout_reg_7/
aluout_reg585/  aluout_reg_2/   aluout_reg_4/   aluout_reg_6/   zero_reg/
```

Related Information

Related command: change_names on page 1301

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_cdn_loop_breaker

```
remove_cdn_loop_breaker
     -instances instance_list design
```

Removes the specified loop breaker buffers added by the timing engine and restores the loop.

Note: As Genus relies on the absence of uncut combinational feedback loops to time the design, invoking the timer through any method (such as `report timing`) will cause the loop breaker cells to be re-inserted unless they are cut by some other means.

Options and Arguments

design
: Specifies the design for which the loop breakers must be removed.

`-instances` *instance_list*
: Specifies the loop breakers instances that you want to remove.

  If this option is not specified, all instance loop breakers will be removed.

Example

The following example first shows the loop breakers inserted, then removes the loop breakers and lists the report again.

```
legacy_genus:/> report cdn_loop_breaker
============================================================
  Generated by:           version
  Generated on:           date
  Module:                 loop
  Technology library:     tutorial 1.1
  Operating conditions:   typical_case (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

   CDN Loop breaker      Driver   Load
----------------------------------------
inst1/cdn_loop_breaker inst1/i1/Y i0/B
legacy_genus:/> remove_cdn_loop_breaker -instance [find / -inst inst1/
cdn_loop_breaker]
legacy_genus:/> report cdn_loop_breaker
============================================================
  Generated by:           Genus Synthesis Solution version
  Generated on:           date
  Module:                 loop
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
  Technology library:     tutorial 1.1
  Operating conditions:   typical_case (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

  No loop breakers to report
```

Related Information

Related command: report cdn_loop_breaker on page 590

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_design

```
reset_design
     [-timing] [-dft]
     [-verbose] [design]
```

Removes all the user-specified timing and DFT objects, as well as all the floorplan objects, and returns all attributes to their default values for the specified design. Alternatively, this command removes every clock domain, cost group, exception, external delay, scan chain, scan segment, test clock domain, and test signal while returning all attributes on the design to their default values. This allows you to reload the design in the same session while debugging some problem and provides faster debugging when repeatedly loading files to debug problems inside one session.

It does not reset the internal state of applications or global settings, so it is not reliable enough to use in a real flow. For example, it does not free collections but only invalidates them. If you try to access collections saved in TCL variables, you will not be able to see those collections. If you have saved pointers to DB objects in TCL variables, they will be invalid and may even cause a crash if used.

If you have any unexpected problems, you should start a new session without using `reset_design`.

Note: This command should be only used for debugging purposes in real implementation flow. Currently, it does not work on MMMC designs.

Options and Arguments

design	Specifies the particular design to reset when there are multiple designs.
`-dft`	Removes only the DFT constraints.
`-timing`	Removes only the timing constraints.
`-verbose`	Prints messages indicating that the command was successful.

Example

- The following examples illustrates that the `reset_design` command has eliminated all external delays in the design:

```
legacy_genus:/designs/phoenix/timing/external_delays> ls
in_1  out_2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/designs/phoenix/timing/external_delays> reset_design phoenix
legacy_genus:/designs/phoenix/timing/external_delays> ls
./
```

Related Information

Related commands:	reset_power_intent on page 1289
Related attribute:	flow_reset_time_after_flow_init

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

rm

`rm object... [-quiet]`

Removes an object from the design hierarchy. This command is similar to its UNIX counterpart.

For a current list of the objects that can be removed, refer to the command help.

If the hierarchical pin or port bus object has a net connection, the net is disconnected first and then the object is removed.

If you remove a design, the CPF-related information will also be removed from the design hierarchy.

Note: The `rm` command does not work on the `pin` or `port` object.

Options and Arguments

object	Specifies the object that you want to remove. Check the command help for a list of the removable object types.
`-quiet`	Suppresses those messages that indicate which objects are being removed. Alternatively, when removing an object, an information message will not be printed.

Examples

- The following example finds the clock objects in the design:

```
legacy_genus:/> find . -clock *
/designs/comp/timing/clock_domains/domain_1/clock1
```

- The following example uses the result of the `find` command to remove the clock. This command also removes all dependent objects. A subsequent `find` cannot find any clock objects.

```
legacy_genus:/> rm [find . -clock *]
Info    : Removing a clock object [TIM-102]
        : The clock name is 'clock'
    Removing external delay 'in_del_1'.
    Removing external delay 'ou_del_1'.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> find . -clock *
I cannot find any clock named * here
Failed on find . -clock *
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

ungroup

```
ungroup
     [ -all | -flatten instance...
     | -threshold integer | instance...]
     [-simple] [-prefix string]
     [-exclude instances]
     [-only_user_hierarchy] [-force] [-verbose]
```

Ungroups the specified instances. Ungrouping dissolves the hierarchy and moves the contents of a subdesign into its parent directory. By default, an instance is named by concatenating its name to its parent's name.

Options and Arguments

`-all`	Ungroups all instances at the current level.
`-exclude` *`instances`*	Specifies a list of instances that should not be ungrouped.
`-flatten`	Recursively ungroups all the specified instances.
`-force`	Forces to ungroup all hierarchies. **Note:** Before mapping, this will include non-user hierarchies.
`instance`	Specifies the instance or instances to be ungrouped.
`-only_user_hierarchy`	Ungroups only those hierarchies created by the user. The hierarchies created by the tool are preserved. This option applies after mapping. **Note:** If you execute this command before you execute the `synthesize -to_mapped` command, ungrouping will by default only apply to user-created hierarchies unless you specify the `-force` option.
`-prefix` *`string`*	Adds the specified prefix to the names of the wires, nets, and instances created as a result of flattening the hierarchical instance(s).
`-simple`	Prevents the use of a complex new instance name during ungrouping. This option has the same effect as the `edit_netlist ungroup` command.
`-threshold` *`integer`*	Ungroups only those hierarchical instances that have a cell count equal to or less than the specified integer.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-verbose`	Prints the names of all hierarchies being ungrouped.

Examples

- The following example ungroups the instance named `CRITICAL_GROUP`:

```
legacy_genus:/> ungroup [find / -instance CRITICAL_GROUP]
```

- In the following example, every instance under the `inst` hierarchical instance will be ungrouped except the `inst_sub2` instance:

```
legacy_genus:/designs/ksable_hier/instances_hier/inst/instances_hier> ls
./      inst_sub1/      inst_sub2/      inst_sub3
legacy_genus:/designs/ksable_hier/instances_hier/
> ungroup inst -flatten -exclude \
    [find . -instance inst/inst_sub2]
```

Related Information

Related command: edit_netlist ungroup on page 1342

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

uniquify

```
uniquify
     {subdesign|design} [-verbose]
```

Uniquifies the instances under the specified design or subdesign. Uniquification is the process of creating a new subdesign for an instance or a group of instances. The newly created subdesign is merely a copy of the subdesign to which the original instance or group of instances were associated. That is, an instance or a group of instances will now be a part of their own, *unique* subdesign. The newly created subdesign will usually maintain the original design or subdesign name followed by a number suffix.

Alias for edit_netlist uniquify.

The naming style of uniquified subdesigns is controlled by the attribute `uniquify_naming_style`.

Note: Preserved modules cannot be uniquified. By default, `uniquify` command will uniquify all modules except its first occurrences, as it will be unique after other modules are uniquified.

Options and Arguments

{*subdesign*|*design*}	The instances under the specified design or subdesigns will be uniquified.
`-verbose`	Prints out the uniquified instances and their corresponding subdesign names.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

17

Joules

Tip

To use the commands below, set the `power_engine` root attribute to `joules`.

- get_lib_domains on page 1362
- report power on page 1363
- report_sdb_annotation on page 1366
- write_icgc_monitor on page 1368

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_lib_domains

```
get_lib_domains
     [-name string] [-invert] [-nocase]
```

Returns the full path to any library domain(s) that matches the specified name.

Options and Arguments

`-nocase`	Indicates that the name is not case sensitive.
`-name` *`string`*	Specifies the name to match with the domain names. Wildcards are supported.
`-invert`	Filters out domains that match the name and return those domains that do not match the given name.

Examples

- `get_lib_domains` # get list of ALL library domains
- `get_lib_domains -name "*1p2*"` # get a LVT/HVT library
- `get_lib_domains -name "1P2" -nocase -invert` # get a LVT/HVT library

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report power

```
report power
     [-stims string...] [-frames string...]
     [-inst hinst...] [-module rtl_module...]
     [-clock_domain clock_domain...]
     [-power_domain power_domain...]
     [-collate {frames|hier|domain|all|none}]
     [-power_mode power_mode] [-power_rail power_rail...]
     [-by_category] [-by_hierarchy]
     [-by_leaf_instance] [-by_rail]
     [-by_func_type] [-by_libcell] [-by_tiles string]
     [-unit {W|mW|uW|nW}] [-format format] [-header]
     [-view analysis_view] [-skip_port_switching_power]
     [> file]
```

Reports the power. Use the options to control the content and format of the report.

Options and Arguments

`-by_category`	Reports a line each for the list of specified design element categories. Where `category` can be any of the following: `{memory | register | latch | logic | bbox | clock | clock_comb | clock_seq | pad | pm}` `pm` stands for power management cells (such as level shifter, retention, isolation cells). Type `report power -by_category -help` for more information on controlling the content and format of the report.
`-by_func_type`	Reports the power by functional type. Type `report power -by_func_type -help` for more information on controlling the content and format of the report.
`-by_hierarchy`	Reports the power by hierarchy. The hierarchy name is always displayed as the rightmost column in the report.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Type `report power -by_hierarchy -help` for more information on controlling the content and format of the report.

`-by_leaf_instance`

Reports the power by leaf instance. This is the equivalent of the Genus legacy command: `report power -flat`

Type `report power -by_leaf_instance -help` for more information on controlling the content and format of the report.

`_by_libcell`

Reports the power by libcell.

Type `report power -by_libcell -help` for more information on controlling the content and format of the report.

`_by_rail`

Reports the power for the specified rails.

`-by_tiles` *`string`*

Reports power by tile.

Where *`string`* is of the following format:

```
rowsXcols [-style {transient | simple}]
[-cols {total | leakage | internal | switching]...
```

Default: `-style transient -col total`

`-clock_domain` *`clock_domain`*

Specifies the clock domains for which to report the power.

`-collate {frames|hier|domain|all|none}`

Collates power reports across frames, or hierarchical instances, or clock/power domains, or all of these.

Default: collates across frames.

`file`

Specifies th e file to which to write the report.

`-format` *`format`*

Specifies the format of report.

`-frames`

Specifies the frames or frame range.

Default: `frame#0` of all stims.

`-inst` *`hinst...`*

Specifies the hierarchical instances for which to report power.

`-header`

Generates a report header.

`-module` *`rtl_module`*

Reports power for all instances of the specified module.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You can control the number of levels to report using `-levels {integer | all}`. The default value is `0`.

`-power_domain` *`power_domain`*

Specifies the power domains for which to report the power.

`-power_mode` *`power_mode`*

Specifies the power modes for which to report the power.

`-power_rail` *`power_rail`*

Specifies the power rails for which to report the power.

`-skip_port_switching_power`

Does not take the port switching power into account.

`-stims` *`string`*

Specifies the stims for which to report the power.

Default: all stims are considered.

`-unit {W|mW|uW|nW}`

Specifies the units to be used to report the power.

Default: `W`

`-view` *`analysis_view`*

Reports the power for a given analysis view.

Examples

- `report power;` # reports power for the top level design(s)
- `report power -frames /stim#1/frame#1 {/stim#1/frame#[5:8]}`
- `report power -module cpu_10bit__fsm`
- `report power -inst /cpu_10bit/FSM -level 1 -unit mW -format %.5f`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_sdb_annotation

```
report_sdb_annotation
     [-stims string...] [-inputs]
     [-show_details obj_type[:annotation_type[:pin_prop_type]]]
     [object]
```

Reports the annotations on a design found in the Joules stimulus database (SDB).

Options and Arguments

`-inputs`
: Reports the annotation of inputs pins. Displays inputs of flops, memories, latches, and integrated clock-gating instances (icgc).

`object`
: Reports the annotation of the specified design or instance.

Default: top design.

`-show_details` *`obj_type`*`[:`*`annotation_type`*`[:`*`pin_prop_type`*`]]`
: Reports annotation details for the specified object, annotation type, and pin type set.

`object_type` can be one of the following:

`{all |port | port_dft | seq | comb | reg | reg_dft | mem | mem_dft | icgc | latch | bbox | pad | pm | mux}`

Default: `all`

`annotation_type` can be one of the following:

- after `read_stimulus` and before `compute_power`

  `{asserted | unasserted | unconnected | all}`

- after `compute_power` or `propagate_activity`

  `{all | asserted | user_asserted | computed | default |clock_source | constant | unasserted |unconnected}`

Default: `all`

`pin_prop_type` can be one of the following:

`{out | in | clock | enable | all}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Default: `out`
`-stims` *`string`*	Reports the annotation of the given stims.
	Default: all stims are considered.

Examples

- `report_sdb_annotation -stims /stim#1 -show_details seq:computed:out`
- `report_sdb_annotation -stims /stim#1 -show_details reg:asserted:clock`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

write_icgc_monitor

```
write_icgc_monitor
     [-out file] [-dut_instance name]
     [-max_expr_rtl_signal_count string]
```

Creates a System Verilog module file for the enable signals of the clock-gating instances that were inserted during synthesis. This file can then be used during simulation to generate the activity on these enable pins.

You must have run generic synthesis before you can use this command.

Important

To run this command, you ill need a `Joules_RTL_Power` or `Joules_Power_SP` license.

Options and Arguments

`-dut_instance name`	Specifies the instance name of the design under test in the testbench.
`-max_expr_rtl_signal_count string`	Specifies the maximum number of RTL signals allowed.
`-out file`	Specifies the name of the output file.

Related Information

Related command: syn_generic on page 526

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

18

Customization

- add_command_help on page 1370
- define_attribute on page 1371
- mesg_make on page 1376
- mesg_send on page 1378
- parse_options on page 1380

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_command_help

`add_command_help` *`command_name help category`*

Adds a short help message for a new command to Genus's online help system. Examples of `add_command_help` can be found in the installation `lib/cdn/genus` directory.

Options and Arguments

`category`	Specifies the category to which this command should be added. You can specify an existing category or a new one. To see a list of all existing categories, type help.
`command_name`	Specifies the name of the command. It must be a Tcl procedure that you defined previously.
`help`	Lists the help text to be displayed when the `help` command is used. Use a string.

Examples

- The following example adds help for the `hello_world` command:

```
legacy_genus:/> proc hello_world {hello_world} { echo "Hello world" }
legacy_genus:/> add_command_help "hello_world" "Says hello to the whole world"
my_category
legacy_genus:/> help hello_world
That command is:

    my_category
================================================================================
hello_world  Says hello to the whole world

Command details:

Hello world
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

define_attribute

```
define_attribute string
     -category string -data_type string -obj_type string
     [-computed |-hidden | obsolete] [-help_string string]
     [-check_function string] [-compute_function string]
     [-name_check_function string]
     [-reset_function string] [-set_function string]
     [-default_value string] [-units string] [-skip_in_db]
```

Creates a new, user-defined attribute with the specified characteristics. These attributes will also be written out if you use the `write_script` command.

Options and Arguments

`-category string`
: Defines the category of the attribute. Categories group attributes that perform similar functions whereas object types describe where in the design an attribute is valid. You can specify any category name: both new and existing category names are valid.

`-check_function string`
: Specifies a previously defined Tcl procedure's name in order to ensure that the newly defined attribute is valid. The Tcl procedure should be of the form:

  `proc {object value}`

  The Tcl procedure returns `1` for a valid value, and `0` for an invalid value.

`-compute_function string`
: Specifies a previously defined Tcl procedure's name in order to get the newly defined attribute's value later (with the `get_attribute`) command. The Tcl procedure should be of the form:

  `proc {object}`

  When you use this option, the attribute becomes a read-only attribute because its value is always computed.

`-computed`
: Marks the attribute as a computed attribute.

  As a result, the attribute will only be shown with the `ls -c` command, and not with the `ls -a` command.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-data_type` *`string`*

Defines the data type of the attribute. Possible data types are:

- bool
- coord
- double
- int
- object
- string

`-default_value` *`string`*

Specifies a default value for the attribute.

`-help_string` *`string`*

Specifies the help text for the attribute.

`-hidden` Specifies whether the defined attribute is a hidden attribute.

`-more_help_string` *`string`*

Specifies an extended help string.

`-name_check_function` *`string`*

Specifies a previously defined Tcl procedure's name to check the value of an attribute. The Tcl procedure should be of the form:

```
proc {object value attribute_name}
```

The Tcl procedure returns `1` for a valid value, and `0` for an invalid value.

`-obj_type` *`string`* Specifies the object type of the attribute. All valid object types can be found by typing `find -help` at the Genus prompt.

`-obsolete` Specifies whether the defined attribute is obsolete.

`-reset_function` *`string`*

Specifies a previously defined Tcl procedure's name. This option allows you to reset a user-defined value. The Tcl procedure should be of the form:

```
proc {object new_value current_value}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-set_function` *`string`*

Specifies a previously defined Tcl procedure's name. This option allows you to set a user-defined value. The Tcl procedure should be of the form:

`proc {`*`object new_value current_value`*`}`

`-skip_in_db` Prevents the `write_db` command to write out the defined attribute.

`string` Specifies the name of the attribute.

`-units` *`string`* Specifies the attribute unit type (for example, seconds, Mbytes, nW)

Examples

- The following example defines a boolean attribute named `new_libpin` for a new category named `my_libpin`:

```
legacy_genus:/> define_attribute -data_type boolean -obj_type libpin -category
my_libpin new_libpin
legacy_genus:/> get_attribute new_libpin * -help
...
attribute category: my_libpin
    attribute name: new_libpin
          category: my_libpin ()
       object type: libpin
       access type: read-write
         data type: boolean
     default value:
              help:
```

- The following example creates a Tcl procedure, `check_fxn`, and then creates an attribute named `test_check`. The `-check_function` option is specified so that the `test_check` attribute can be tested for validity when it is specified later.

```
legacy_genus:/> proc check_fxn {obj val} {
==>   if {$val < 0} {
==>     return 0
==>   }
==>   return 1
==> }

legacy_genus:/> define_attribute test_check -obj_type root -data_type integer
\
    -category test -help_string "test check function" \
    -check_function check_fxn
/object_types/root/attributes/test_check
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The following command is a valid use of the newly created `test_check` attribute:

```
legacy_genus:/> set_attribute test_check 1 /
Setting attribute of root '/': 'test_check' = 1
```

The following command would be an invalid use:

```
legacy_genus:/> set_attribute test_check -1 /
Error  : The data value for this attribute is invalid. [TUI-24] [set_attribute]
       : The value '-1' cannot be set for attribute 'test_check'.
       : To see the usage/description for this attribute, type 'set_attribute
-h <attr_name> *'.
```

- The following example creates a Tcl procedure, `set_fxn`, and then creates an attribute named `test_set`. The `-set_function` option is specified so that you can change the value of the `test_set` attribute (provided it is valid):

```
legacy_genus:/> proc set_fxn {obj new_val cur_val} {
==>   if {$new_val > $cur_val} {
==>     return $new_val
==>   }
==>   return $cur_val
==> }
legacy_genus:/> define_attribute test_set -obj_type root -data_type integer
-units bytes\
    -category test -help_string "test set function" -set_function set_fxn

/object_types/root/attributes/test_set
```

The `test_set` attribute will be changed to `1`. It is valid, since there was no previous value:

```
legacy_genus:/> set_attribute test_set 1
Setting attribute of root '/': 'test_set' = 1
legacy_genus:/> get_attribute test_set * -h

Usage: get_attribute <string> [<object>+]

    <string>:
        attribute name
    [<object>+]:
        object of interest (must be unique)

    attribute category: test

    attribute name: test_set
          category: test ()
       object type: root
       access type: read-writ
         data type: integer
     default value:
             units: bytes
              help: test set function
```

The following command changes the attribute value to `2`. Again, this is valid because it falls within the definition of the previously defined Tcl procedure, `set_fxn`:

```
legacy_genus:/> set_attribute test_set 2 /
Setting attribute of root '/': 'test_set' = 2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The attribute's value will not be changed in the following example. The value will remain at `2`:

```
legacy_genus:/> set_attribute test_set 0 /
Setting attribute of root '/': 'test_set' = 2
```

- The following example creates a Tcl procedure, `compute_fxn`, which always returns the value of 42. The `define_attribute` command then creates an attribute named `test_compute`. The `-compute_function` option is specified so that you can obtain the `test_compute` attribute's value later:

```
legacy_genus:/> proc compute_fxn {obj} {
==>   return 42
==> }
legacy_genus:/> define_attribute test_compute -obj_type root -data_type
integer \
    -category test -help_string "test compute function" \
    -compute_function compute_fxn -computed

/object_types/root/attributes/test_compute
```

The `test_compute` value will always be `42`, as defined in the Tcl procedure:

```
legacy_genus:/> get_attribute test_compute /
42

legacy_genus:/> set_attribute test_compute 23 /
Error   : The attribute is read-only. [TUI-26] [set_attribute]
        : attribute: 'test_compute', object type: 'root'
        : Cannot set or reset read-only attributes.
Failed on set_attribute test_compute 23
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

mesg_make

```
mesg_make -group string [-internal_group] -id number
     -short_description string
     -long_description string
     {-error|-warning|-info_priority number}
```

Creates a custom message that can subsequently be accessed with the `mesg_send` command.

Options and Arguments

`-error`	Creates an error message.
`-group` *`string`*	Specifies the group of messages that the new message belongs to. A group groups messages that apply to a certain engine of the tool. For example, the MAP group groups messages issued by the mapper. **Note:** If you want to create a message for an internal group, you must specify an existing group name.
`-id` *`integer`*	Specifies an identification number for the message. The number must be unique for the specified group. If the specified number already exists, you will overwrite the existing message. *Default*: `1`
`-info_priority` *`integer`*	Creates an info message with the specified priority. You can specify a number between 2 and 8.
`-internal_group`	Specifies whether the message belongs to an internal group.
`-long_description` *`string`*	Defines the help of the message.
`-short_description` *`string`*	Specifies the title or the description of the message.
`-warning`	Creates a warning message.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following example creates a message in the `test` group and assigns it a unique identification number (501) within that group:

```
legacy_genus:/> mesg_make -group test  -id 501 -short_desc note_bene \
==> -long_desc "search for lost time" -warning
/messages/test/test-501
legacy_genus:/> man test-501
Entry       : test-501
Severity    : Warning
Verbosity   : Message is visible at any 'information_level' above '1'.
Description : note_bene
Help        : search for lost time
```

Related Information

Affects this command: mesg_send on page 1378

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

mesg_send

`mesg_send` *`message string`*

Accesses the various native messages of Genus and the messages that were created with the `mesg_make` command.

Options and Arguments

`-caller` *`string`*	Specifies the name of the calling procedure.
`-file_info` *`string`*	Specifies to which file the message applies.
`message`	Identifies the message to be sent. The identification is in the form of *`group`*-*`id`*. Refer to mesg_make for an explanation of *`group`* and *`id`*
`-newline`	Prints a new (empty) line before the message
`-object_info` *`string`*	Prints the object type and
`string`	Describes the context-specific help.

Examples

- The following example accesses the message created in Example on page 1377 for `mesg_make`:

```
legacy_genus:/messages/test> mesg_send test-501 "reminders" -newline

Warning : note_bene [test-501]
        : reminders
        : search for lost time
```

- The following example sends a message after elaboration:

```
legacy_genus:/> mesg_send /messages/VHDLPT/VHDLPT-500 "file not found" -caller
read_hdl
Error   : Cannot open file. [VHDLPT-500] [read_hdl]
        : file not found
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Affected by this command: mesg_make on page 1376

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

parse_options

`parse_options` *`cmd file_var`* [*`args`*] [*`code var`*]...

Interfaces to the Genus internal command option parser. The Genus argument parser provides the following features:

- Checking the correctness for arguments and types. Appropriate messages are issued when the input is incorrect.
- No specific order of arguments is required. For example, `ls -long -attribute` behaves just like `ls -attribute -long`.
- Unique abbreviations of arguments is supported. For example, `ls -l` behaves just like ls `-long`.
- Online help is provided if `-help` is specified.
- Optional file redirection is supported. For example, `report gates >> design.rpt` causes output to be appended to the file `design.rpt`.
- Genus objects can be implicitly searched for based on their type. For example, `fanout SUB/A[0]` performs an implicit find on the string `SUB/A[0]` and locates the object `/designs/TOP/instances_hier/SUB/pins_in/A[0]`.

You can use the command option parser to make a Tcl procedure behave just like a built-in Genus command by providing on-line help, finding objects automatically, checking for required options, handling unique argument abbreviations, and handling file redirection.

This command can return one of the following values:

- -2: You asked for help and help was provided. The command returns normally.
- 0: Your options were invalid and the command aborts.
- 1: Your options were valid and the command continues normally.

Options and Arguments

`args`

Specifies a list containing the options that the user sends to your command. Usually your procedure will be defined like this:

```
proc your_procedure {args} {
...  (code that implements the procedure)
}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

You would then pass `$args` as the args parameter to `parse_options` within your procedure.

cmd

Specifies the name of the command whose options to parse.

This name appears in the help listing if a user calls your procedure with `-h` or if a user does not provide valid arguments.

code

Specifies a string that describes the command argument: flag name, whether it is required or optional, what type of data it accepts, and a short help message about it. The string must be in the form:

```
"(-<name>)?<x><y><z>(<dirtypes>)? <help>"
```

The question marks in the above string mean that these fields of the string are optional.

You cannot specify multiple string values if you are specifying a flag name. That is, if you are specifying a flag, you cannot also specify the `som` (string, optional, multiple values) or `srm` (string, required, multiple values) combinations.

<name> is the name of the flag. For example, `-number`.

<x> is a single character indicating the type of the option:

`b`	Boolean
`d`	directory object
`e`	enum
`f`	float
`n`	integer number
`s`	string

<y> is a single character indicating whether the option is optional or required

`o`	optional
`r`	required
`x`	obsolete

<z> is a single character indicating whether lists are accepted

`m`	Accepts multiple values: lists OK
`s`	Accepts single value only: no lists

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	<dirtypes> is a string indicating the types of directory objects the option accepts. This string is required for all arguments that have the *<x>* field set to `d` and cannot be specified otherwise. The list of specified directory types can only be separated by vertical bars (|), that is, no spaces allowed, and must be enclosed in parentheses. *<help>* is a string that indicates to the user of your command what the purpose of the option is.
file_var	Specifies the name of a variable that holds the file handle if the user calls your procedure with > *file* or >> *file*. Your procedure should always check to see if this variable has been set to something other than 'stdout'. If it has, then your command should send its output to that file handle instead of 'stdout' and you should close the file handle once your command is complete.
var	Specifies the name of the variable that will be set with the parsed result for that argument.

Examples

- The following example shows how file indirection works:

```
legacy_genus:/> parse_options hello_world file_var {> ./tmp}
1
legacy_genus:/> puts $file_var "Hello world!"
legacy_genus:/> close $file_var
```

- The following example shows a required argument with the `-design` flag that must be the name of a subdesign (block).

```
legacy_genus:/> parse_options hello_world file_var {-design addinc65} \
==> "-design drs(subdesign) A module"   my_design
1
legacy_genus:/> puts $my_design
/designs/alu/subdesigns/addinc65
```

- The following example shows a Boolean argument, a numeric argument, and an un-flagged object argument that can be either a subdesign (block) or an instance. It also shows how the parser accepts abbreviations since the argument passed in is only `-t`, it still gets recognized as `-top`.

```
legacy_genus:/> set level 300
300
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> parse_options hello_world file_var {-t addinc65} \
==>  "-top bos Do the top level thing"  top \
==>  "-level nos The level to work on"  level \
==>  "dos(subdesign|instance) An object to work on"   object
1

legacy_genus:/> puts $object
/designs/alu/subdesigns/addinc65

legacy_genus:/> puts $top
1

legacy_genus:/> puts $level
300
```

- The following example shows how online help works:

```
legacy_genus:/> parse_options hello_world file_var {-h} \
==>  "-top bos Do the top level thing"  top \
==>  "-level nos The level to work on"  level \
==>  "dos(subdesign|instance) An object to work on"   object
 Usage: hello_world [-h]
          -h: this message
        hello_world [-top] [-level number] [instance|subdesign] [> file]
          -top: (Boolean) Do the top level thing
          -level:       (integer) The level to work on
                        (instance|subdesign) An object to work on
-2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

19

SDC Commands

- Introduction on page 1391
- add_to_collection on page 1392
- all_clocks on page 1395
- all_connected on page 1396
- all_fanin on page 1397
- all_fanout on page 1399
- all_inputs on page 1401
- all_instances on page 1402
- all_outputs on page 1403
- all_registers on page 1404
- append_to_collection on page 1406
- compare_collections on page 1408
- copy_collection on page 1409
- create_clock on page 1410
- create_generated_clock on page 1412
- current_design on page 1416
- current_design_name on page 1417
- current_instance on page 1418
- filter_collection on page 1419
- foreach_in_collection on page 1421
- get_arcs on page 1422

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- get_cells on page 1424
- get_clocks on page 1428
- get_designs on page 1430
- get_generated_clocks on page 1432
- get_leaf_cells on page 1435
- get_leaf_nets on page 1436
- get_leaf_pins on page 1437
- get_libs on page 1438
- get_lib_arcs on page 1440
- get_lib_cells on page 1443
- get_lib_pins on page 1445
- get_nets on page 1448
- get_object_name on page 1451
- get_path_groups on page 1452
- get_pins on page 1454
- get_ports on page 1458
- get_references on page 1461
- getenv on page 1462
- group_path on page 1463
- index_collection on page 1467
- list_property on page 1468
- load_of on page 1469
- remove_case_analysis on page 1470
- remove_clock_gating_check on page 1471
- remove_clock_latency on page 1472
- remove_clock_transition on page 1473
- remove_disable_clock_gating_check on page 1474

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- remove_disable_timing on page 1476
- remove_from_collection on page 1477
- remove_generated_clock on page 1479
- remove_ideal_net on page 1480
- remove_ideal_network on page 1481
- remove_input_delay on page 1482
- remove_output_delay on page 1483
- report_property on page 1484
- reset_clock on page 1485
- reset_clock_gating_check on page 1486
- reset_clock_groups on page 1487
- reset_clock_latency on page 1488
- reset_clock_sense on page 1489
- reset_clock_transition on page 1490
- reset_clock_uncertainty on page 1491
- reset_data_check on page 1493
- reset_disable_clock_gating_check on page 1495
- reset_disable_timing on page 1496
- reset_drive on page 1497
- reset_driving_cell on page 1498
- reset_generated_clock on page 1499
- reset_ideal_network on page 1500
- reset_input_delay on page 1501
- reset_load on page 1503
- reset_max_capacitance on page 1504
- reset_max_fanout on page 1505
- reset_max_time_borrow on page 1506

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- reset_max_transition on page 1507
- reset_min_capacitance on page 1508
- reset_min_fanout on page 1509
- reset_min_pulse_width on page 1510
- reset_min_transition on page 1511
- reset_mode on page 1512
- reset_output_delay on page 1513
- reset_path on page 1515
- reset_path_group on page 1517
- reset_resistance on page 1518
- reset_timing_derate on page 1519
- reset_wire_load_mode on page 1521
- reset_wire_load_model on page 1522
- reset_wire_load_selection_group on page 1523
- set_active_clocks on page 1524
- set_annotated_check on page 1525
- set_annotated_delay on page 1527
- set_annotated_transition on page 1530
- set_case_analysis on page 1531
- set_clock_exclusive on page 1532
- set_clock_gating_check on page 1533
- set_clock_groups on page 1536
- set_clock_latency on page 1539
- set_clock_sense on page 1543
- set_clock_skew on page 1545
- set_clock_transition on page 1547
- set_clock_uncertainty on page 1548

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- set_constraint_mode on page 1551
- set_data_check on page 1552
- set_disable_clock_gating_check on page 1556
- set_disable_timing on page 1558
- set_dont_touch on page 1560
- set_dont_touch_network on page 1561
- set_dont_use on page 1562
- set_drive on page 1563
- set_driving_cell on page 1566
- set_equal on page 1570
- set_false_path on page 1571
- set_fanout_load on page 1577
- set_hierarchy_separator on page 1578
- set_ideal_latency on page 1579
- set_ideal_net on page 1580
- set_ideal_network on page 1581
- set_input_delay on page 1582
- set_input_transition on page 1586
- set_lib_pins on page 1588
- set_load on page 1590
- set_load_unit on page 1592
- set_logic_dc on page 1593
- set_logic_one on page 1594
- set_logic_zero on page 1595
- set_max_capacitance on page 1596
- set_max_delay on page 1598
- set_max_dynamic_power on page 1603

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- set_max_fanout on page 1604
- set_max_leakage_power on page 1605
- set_max_time_borrow on page 1606
- set_max_transition on page 1608
- set_min_capacitance on page 1610
- set_min_delay on page 1612
- set_min_fanout on page 1615
- set_min_pulse_width on page 1616
- set_min_transition on page 1618
- set_mode on page 1619
- set_multicycle_path on page 1621
- set_operating_conditions on page 1625
- set_opposite on page 1627
- set_output_delay on page 1628
- set_path_adjust on page 1632
- set_port_fanout_number on page 1635
- set_resistance on page 1636
- set_size_only on page 1637
- set_timing_derate on page 1638
- set_time_unit on page 1641
- set_unconnected on page 1642
- set_units on page 1643
- set_wire_load_mode on page 1645
- set_wire_load_model on page 1646
- set_wire_load_selection_group on page 1648
- sizeof_collection on page 1650
- sort_collection on page 1651

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Introduction

Currently this appendix describes the new commands added in SDC 1.7 and some other frequently used SDC commands.

SDC commands can be specified

- in an SDC file and read in with the `read_sdc` command
- at the Genus prompt

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

add_to_collection

```
add_to_collection
     base_collection second_collection [-unique]
```

Creates a new collection by concatenating objects from the base collection and the second collection.

- If all the objects in the base collection have the *same* type, only those objects in the second collection with the same type as those in the base collection are added in the result collection. If you specify an object list in second collection, the command converts the objects list to a single list and will add the objects with the same type as those in the base collection to result collection.
- If the objects in the base collection have *different* types, all objects in the second collection are added to the result collection.
- If the second collection is empty, the new collection will be a copy of the base collection.
- If the base collection is empty, the new collection will be a copy from the second collection.

If the software finds no objects to add, an empty collection is returned.

Options and Arguments

base_collection	Specifies the base collection of objects. Objects from the second collection or object list are added to this collection to create the new collection. **Note:** The base collection cannot contain an object list.
second_collection	Specifies the second collection or a list of objects.
-unique	Removes the duplicate objects from the new collection. If this option is omitted, duplicate objects will not be removed.

Examples

Assume you define the following variables:

```
legacy_genus:/> set coll_inst [get_cells *]
/designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
legacy_genus:/> set coll_port [get_ports *]
/designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/ports_in/
sel /designs/test/ports_out/out1
legacy_genus:/> set obj_list {/designs/test/instances_seq/U3 /designs/test/
instances_comb/U2/pins_out/Z {/designs/test/ports_in/clk1 /designs/test/ports_in/
data1 /designs/test/ports_in/sel /designs/test/ports_out/out1}}
```

- The new collection created by the following command is identical to the base collection because the all objects of the base collection are instances while the objects of the second collection are ports:

```
legacy_genus:/> add_to_collection $coll_inst $coll_port
/designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2
```

- The new collection created by following command contains all objects from both collections because the types of the objects of the base collection are different. Note that instance U3 is added twice.

```
legacy_genus:/> add_to_collection $obj_list1 $coll_inst
/designs/test/instances_seq/U3 /designs/test/instances_comb/U2/pins_out/Z {/
designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/
ports_in/sel /designs/test/ports_out/out1} /designs/test/instances_seq/U3 /
designs/test/instances_comb/U0 /designs/test/instances_comb/U1 /designs/test/
instances_comb/U2
```

- The following collection will have one object less than the previous one because only unique objects are collected.

```
legacy_genus:/> add_to_collection $obj_list1 $coll_inst -unique
/designs/test/instances_seq/U3 /designs/test/instances_comb/U2/pins_out/Z {/
designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/
ports_in/sel /designs/test/ports_out/out1} /designs/test/instances_comb/U0 /
designs/test/instances_comb/U1 /designs/test/instances_comb/U2
```

- The new collection created by the following command is identical to the base collection because the all objects of the base collection are instances while the second collection contains the same instances and the design. Consequently the design is not added to the collection, and because the -unique option was given, only the instances of the base collection are added to the resulting collection.

```
legacy_genus:/> add_to_collection [get_cells *] \
[concat [get_cells *] [get_designs *]] -unique
/designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2
```

Related Information

Related commands: append_to_collection on page 1406
compare_collections on page 1408

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

copy_collection on page 1409

remove_from_collection on page 1477

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_clocks

`all_clocks`

Returns all clocks in the design when you do not perform multi-mode timing analysis. This command creates a collection of all the clocks based on the defined or created clocks in the design. If you do not define or create any clocks, an empty string is returned.

When performing multi-mode timing analysis, the command returns all clocks that apply to the current mode.

- When you use this constraint in an SDC file, the constraint will be applied to the mode specified with the `-mode` option of the read_sdc command that reads in the file.
- When you want to use this constraint at the Genus prompt, make sure that the mode to which this constraint needs to be applied was previously specified using the set_interactive_constraint_modes command.
- f no clocks were defined, an empty string "" is returned.
- To filter clocks, use the `get_clocks` command to create a collection of the clocks matching a specific pattern. Assign these clocks to a variable or pass them using another command.

Related Information

Related commands:

all_inputs on page 1401
create_clock on page 1410
get_clocks on page 1428

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_connected

`all_connected [-leaf_pin] { net | pin | pg_pin | port}...`

If the specified object is a net, the command returns the list of all the pins and ports connected to the net. If the object is a pin or a port, the command returns the net connected to the pin or the port.

This command accepts a TCL list as an argument but not wildcards.

Options and Arguments

`-leaf_pin`	Returns the leaf cell pins and the ports connected to the specified net.
net	Specifies a net. The ensuing list will return a list of all the pins and ports connected to this net.
pin	Specifies a pin. The ensuing list will return the net connected to the pin.
pg_pin	Specifies a power or ground pin. The ensuing list will return the net connected to the pin.
port	Specifies a port. The ensuing list will return the net connected to the port.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_fanin

```
all_fanin -to {pin|port|net|pin_bus|port_bus}...
     [-flat] [-levels integer]
     [-only_cells] [-startpoints_only]
     [-trace_through {case_disable |user_disable | all}]
     [-view view_name]
```

Returns a list of pins, ports, or cells that exist in the fanin cone of the specified objects.

Options and Arguments

`-flat`
: Returns all objects in the fanin cone of the specified objects, even when they are present in a hierarchy above the specified objects.

`-levels integer`
: Specifies the number of logic levels to go back to report the fanin cone information.

`-only_cells`
: Returns only the instance names of all objects in the fanin cone.

`-startpoints_only`
: Returns only the timing start points in the fanin cone.

`-trace_through {case_disable |user_disable | all}`
: You can specify one of the following options:

  `case_disable`—Allows traversal of arcs disabled using `set_case_analysis` command. Arcs disabled using set_disable_timing are not traversed.

  `user_disable`—Allows traversal of arcs disabled using `set_disable_timing` command, but arcs disabled using set_case_analysis command are not traversed.

  `all`—Allows traversal through all the arcs.

`-to {pin | port | net | pin_bus | port_bus}`
: Specifies a list of pins, ports or nets for which you want the fanin cone information.

`-view view_name`
: Specifies the view name for which the fanin cone traversal must be performed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

The following module is used for the examples:

```
module top(in, out, clk);
    input in, clk;
    output out;
    wire in, clk;
    wire out;
    wire w1, w2;
    fflopd inst1(.CK (clk), .D (in), .Q (w1));
    inv1 inst2(.A (w1), .Y (w2));
    fflopd inst3(.CK (clk), .D (w2), .Q (out));
endmodule
```

- The following command returns the fanin of pin `A` of instance `inst2`.

  ```
  legacy_genus:/> all_fanin -to inst2/A
  ```

  ```
  /designs/top/instances_comb/inst2/pins_in/A /designs/top/instances_seq/inst1/
  pins_out/Q /designs/top/instances_seq/inst1/pins_in/CK
  ```

- The following command returns the timing start points in the fanin cone of pin `A` of instance `inst2`.

  ```
  legacy_genus:/> all_fanin -to inst2/A -startpoints_only
  ```

  ```
  /designs/top/instances_seq/inst1/pins_in/CK
  ```

- The following command returns the instances in the fanin cone of pin `A` of instance `inst2`.

  ```
  legacy_genus:/> all_fanin -to inst2/A -only_cells
  ```

  ```
  /designs/top/instances_comb/inst2 /designs/top/instances_seq/inst1
  ```

- The following command returns the fanin cone of pin `D` of instance `inst3`.

  ```
  legacy_genus:/> all_fanin -to inst3/D
  ```

  ```
  /designs/top/instances_seq/inst3/pins_in/D /designs/top/instances_comb/inst2/
  pins_out/Y /designs/top/instances_comb/inst2/pins_in/A /designs/top/
  instances_seq/inst1/pins_out/Q /designs/top/instances_seq/inst1/pins_in/CK
  ```

- The following command reports only one level in the fanin cone of pin `D` of instance `inst3`.

  ```
  legacy_genus:/> all_fanin -to inst3/D -levels 1
  ```

  ```
  /designs/top/instances_seq/inst3/pins_in/D /designs/top/instances_comb/inst2/
  pins_out/Y /designs/top/instances_comb/inst2/pins_in/A
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_fanout

```
all_fanout
     {-from {pin|port|net}... |-clock_tree}
     [-flat [-levels integer]] [-pin_levels integer]
     [-only_cells] [-endpoints_only] [-stop_at_seqs]
     [-trace_through {case_disable |user_disable | all}]
     [-view view_name]
```

Returns a list of pins, ports, or cells that exist in the fanout cone of the specified objects.

Options and Arguments

`-clock_tree`
Returns the objects in the fanout of the clock network.

`-endpoints_only`
Returns only the timing end points in the fanout cone.

`-flat`
Returns all objects in the fanout cone of the specified objects, even when they are present in a hierarchy above the specified objects.

`-from {pin | port | net}`
Specifies a list of pins, ports or nets for which you want the fanout cone information.

`-levels integer`
Specifies the number of logic levels to go back to report the fanout cone information.

This option can only be specified with the `-flat` option.

`-only_cells`
Returns only the instance names of all objects in the fanout cone.

`-pin_levels integer`
Specifies the number of logic levels to go back to report the fanout cone information. It returns all objects in the fanout cone of the sink list that are present within this many pin levels of the sink object.

`-stop_at_seqs`
Stop the traversal at a pin if the next arc is a clock_edge arc.

`-trace_through {case_disable |user_disable | all}`
You can specify one of the following options:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	`case_disable`—Allows traversal of arcs disabled using `set_case_analysis` command and physical constants. Arcs disabled using `set_disable_timing` are not traversed. `user_disable`—Allows traversal of arcs disabled using `set_disable_timing` command, but arcs disabled using `set_case_analysis` command are not traversed. `all`—Allows traversal through all the arcs.
`-view` *`view_name`*	Specifies the view name for which the fanout cone traversal must be performed.

Examples

The following module is used for the examples:

```
module top(in, out, clk);
    input in, clk;
    output out;
    wire in, clk;
    wire out;
    wire w1, w2;
    fflopd inst1(.CK (clk), .D (in), .Q (w1));
    inv1 inst2(.A (w1), .Y (w2));
    fflopd inst3(.CK (clk), .D (w2), .Q (out));
endmodule
```

- The following command returns the fanout of pin `CK` of instance `inst1`.

  ```
  legacy_genus:/> all_fanout -from inst1/CK
  ```

  ```
  /designs/top/instances_seq/inst1/pins_in/CK /designs/top/instances_seq/inst1/
  pins_out/Q /designs/top/instances_comb/inst2/pins_in/A /designs/top/
  instances_comb/inst2/pins_out/Y /designs/top/instances_seq/inst3/pins_in/D
  ```

- The following command returns only the timing end points in the fanout cone of pin `CK` of instance `inst1`.

  ```
  legacy_genus:/> all_fanout -from inst1/CK -endpoints_only
  ```

  ```
  /designs/top/instances_seq/inst3/pins_in/D
  ```

- The following command returns only the instances in the fanout cone of pin `CK` of instance `inst1`.

  ```
  legacy_genus:/> all_fanout -from inst1/CK -only_cells
  ```

  ```
  /designs/top/instances_seq/inst1 /designs/top/instances_comb/inst2 /designs/
  top/instances_seq/inst3
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_inputs

```
all_inputs
     [-clock clock_list [-edge_triggered | -level_sensitive]]
     [-clock_domains clock_domain_list
        [ -edge_triggered | -level_sensitive | -no_clocks]]
```

Returns all the input ports in the design. Using the command options you can filter the ports.

Options and Arguments

`-clock` *`clock_list`*

Returns all input ports that have an input delay relative to the specified clock(s).

`-clock_domains` *`clock_domain_list`*

Returns all input ports that have an input delay relative to the clocks in the specified clock_domain(s).

`-edge_triggered`

Returns all input ports that have a non level-sensitive input delay with respect to the specified clocks.

If the `-clock` option is omitted, it returns all input ports that have an edge-triggered input delay with respect to any clock in the design.

`-level_sensitive`

Returns all input ports that have a level-sensitive input delay with respect to the specified clocks.

If the `-clock` option is omitted, it returns all input ports that have a level-sensitive input delay with respect to any clock in the design.

`-no_clocks`

Returns all input ports that do not have a clock definition.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_instances

```
all_instancess
     {design | libcell | subdesign}
```

Returns a list of instances available in the design, or all instances of the specified libcell or subdesign. The search for instances is made relative to the current instance within the current design.

Options and Arguments

design	Returns the instances of the specified design. You cannot use this command at the top level of the design. You can use it for hierarchical modules below the top design.
libcell	Returns the instances of the specified library cell.
subdesign	Returns the instances of the specified subdesign.

Examples

- The following command returns the instances of design `m3`:

```
legacy_genus:/> ls subdesigns
/designs/m1/subdesigns:
./          m2/         m2myclk/    m3/         m3_0/       m3_0_0/     m4/
m5/         m5_bbox/    m6/
legacy_genus:/> all_instances [get_designs m3]
/designs/m1/instances_hier/m2/instances_hier/m3
```

- The following command returns the instances of libcell `CLKINVX2`

```
legacy_genus:/> all_instances [get_libcells CLKINVX2]
/designs/m1/instances_comb/g127 /designs/m1/instances_comb/g132 /designs/m1/
instances_hier/m2/instances_comb/g150 /designs/m1/instances_hier/m2/
instances_hier/m3/instances_hier/m4/instances_hier/m5/instances_comb/g58
legacy_genus:/>
```

- The following command returns the instances of subdesign `m4`.

```
legacy_genus:/> all_instances /designs/m1/subdesigns/m4
/designs/m1/instances_hier/m2/instances_hier/m3/instances_hier/m4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_outputs

```
all_outputs
     [-clock clock_list | -clock_domains clock_domain_list]
     [-edge_triggered | -level_sensitive]
```

This command creates a collection of all output ports in the current design. To filter outputs, use the get_ports command to create a collection of ports matching certain criteria. Assign these ports to a variable or pass them using another command. You can limit the contents of the collection to ports with output delay relative to a specific clock.

Options and Arguments

`-clock` *`clock_list`*

Returns all output ports that have an output delay relative to the specified clocks

`-clock_domains` *`clock_domain_list`*

Returns all output ports that have an output delay relative to the clocks in the specified clock_domain(s).

`-edge_triggered`

Returns all output ports that have a non level-sensitive output delay with respect to the specified clocks

If you omit the -clock option, the command returns all output ports that have a non level-sensitive output delay to any clock in the design.

`-level_sensitive`

Returns all output ports that have a level-sensitive output delay with respect to the specified clocks.

If you omit the -clock option, the command returns all output ports that have a level-sensitive output delay to any clock in the design.

Related Information

Related commands:

get_ports on page 1458

set_output_delay on page 1628

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

all_registers

```
all_registers
     [-cells]
     [-clock clock...]
     [-fall_clock clock...] [-rise_clock clock...]
     [-clock_domains clock_domain...]
     [-data_pins] [-clock_pins] [-slave_clock_pins]
     [-async_pins] [-output_pins]
     [-flops | -edge_triggered]
     [-latches | -level_sensitive]
     [-macros] [-master_slave] [-inverted_output]
     [-exclude {instance}...]
     [-no_hierarchy]
```

Returns all the sequential elements in the design. Using the command options you can extract specific subsets of elements or specific object types.

Note: This command was added in SDC 1.7.

Options and Arguments

`-async_pins`
: Returns the asynchronous pins of all sequential elements in the design.

`-cells`
: Creates a collection of cells. This is the default option.

`-clock clock_list`
: Returns only those sequential elements clocked by the specified clocks.

`-clock_domains clock_domain_list`
: Returns only those sequential elements that belong to the specified clock domains.

`-clock_pins`
: Returns the clock pins of all sequential elements in the design.

`-data_pins`
: Returns the synchronous pins of all sequential elements in the design.

`-exclude {inst|hinst}`
: Excludes the specified instances from the list of sequential elements to be returned.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall_clock` *`clock_list`*

Returns only those sequential elements clocked by the falling edge of the specified clocks.

`-flops | -edge_triggered`

Returns all sequential elements of type flop in the design.

`-inverted_output` Returns all sequential elements in the design that have an inverted output.

`-latches | -level_sensitive`

Returns all sequential elements of type latch in the design.

`-macros` Returns only macros.

`-master_slave` Returns all sequential elements in the design that are master-slave flops.

`-no_hierarchy` Returns only those sequential elements at the top-level of the design.

`-output_pins` Returns the output pins of all sequential elements in the design.

`-rise_clock` *`clock_list`*

Returns only those sequential elements clocked by the rising edge of the specified clocks.

`-slave_clock_pins` Returns the clock pins of the slave flops in the design.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

append_to_collection

```
append_to_collection
     base_collection_name second_collection
     [-unique]
```

Appends objects from one collection to another collection.

If the objects in the base collection have the *same* type, only those objects in the second collection with the same type as those in the base collection are appended to the base collection. If you specify an object list as the second collection, the command converts the objects list to a single list and appends the objects with the same type as those in the base collection to the base collection.

If the objects in the base collection have *different* types, all objects in the second collection are appended to the base collection. If the second collection has an object list than the command converts the object list to a single collection and append the objects to the base collection for this scenario.

If the base collection is empty than the result collection is the second collection. If the second collection is empty than the result collection is the base collection.

Options and Arguments

base_collection_name

Specifies the variable name of (or pointer to) the base collection to which the objects of the second collection must be added. This base collection becomes the result collection.

Note: The base collection cannot contain an object list.

Note: If this argument is a collection instead of a name, its content will be ignored and the result collection will be the second collection.

second_collection Specifies the second collection.

-unique Removes the duplicate objects from the new collection.

If this option is omitted, duplicate objects will not be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- In the following example, the base collection (referred to by variable `des_name`) contains objects of type design, while the second collection contains objects of type port. Consequently the resulting collection is the base collection.

```
legacy_genus:/> set des_name [get_designs *]
/designs/test
legacy_genus:/> append_to_collection des_name [get_ports *]
/designs/test
```

- In the following example, the first argument is `$des_name`, which is not the collection name, but the collection itself. In this case, the resulting collection is the second collection.

```
legacy_genus:/> set des_name [get_designs *]
/designs/test
legacy_genus:/> append_to_collection $des_name [get_ports *]
/designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/
ports_in/sel /designs/test/ports_out/out1
```

- In the following example, the first argument is also a collection. So, the resulting collection is the second collection.

```
legacy_genus:/> append_to_collection [get_designs *] [get_ports *]
/designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/
ports_in/sel /designs/test/ports_out/out1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

compare_collections

```
compare_collections
     first_collection second_collection
     [-order_dependent]
```

Compares the contents of two collections. If the objects in both collections are the same, the command returns 0, otherwise the command returns 1.

Options and Arguments

first_collection	Specifies the name of the first collection of objects.
-order_dependent	Specifies that the order of the objects in both collections must be the same in order for the collections to be considered the same.
second_collection	Specifies the name of the second collection of objects.

Examples

- The following command compares two collections without considering the order:

```
legacy_genus:/> compare_collections [concat [get_cells *] [get_ports *]] \
==> [concat [get_ports *] [get_cells *]]
1
```

- The following command compares the same collections this time considering the order:

```
legacy_genus:/> compare_collections [concat [get_cells *] [get_ports *]] \==>
[concat [get_ports *] [get_cells *]] -order_dependent
0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

copy_collection

```
copy_collection
     collection
```

Specifies the collection to be copied. If an empty collection is specified than the command returns an empty collection.

Options and Arguments

collection	Specifies the name of a collection of objects to be copied.

Example

- The following command copies collection in variable l1.

```
legacy_genus:/> set l1 {clk d sena clr apre srl srd q}
clk d sena clr apre srl srd q
legacy_genus:/> copy_collection $l1
clk d sena clr apre srl srd q
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_clock

```
create_clock
     [-add]
     [-name clock] [-domain clock_domain]
     -period float [-waveform float]
     [-apply_inverted {port|pin}]
     [port|pin] [-comment string]
```

Creates a clock object and defines its waveform. If you do not specify any sources, but you specify the `-name` option, a virtual clock is created.

Options and Arguments

`-add`
: Allows to specify multiple clocks on the same source for simultaneous analysis with different clock waveforms.

  If you omit this option and a clock was already defined on a pin or port, this definition would overwrite the previous one.

`-apply_inverted {port | pin}`
: Specifies inverted sources of the clock.

  Note: This is non-SDC option.

`-comment string`
: Specifies a comment to be tagged to this command.

`-domain clock_domain`
: Specifies the clock domain to which the clock belongs.

  Default: `domain_1`

  Note: This is non-SDC option.

`-name clock`
: Specifies the name of the clock. The specified name must be unique. If a clock with this name was already defined, the clock definition will be replaced.

  If you omit this option, the clock gets the same name as the first clock source (inverted or non-inverted). If no sources were specified for this clock, the clock name defaults to `create_clock_n`, where `n` is 1,2, and so on.

`-period float`
: Specifies the length of the clock period.

`{port|pin}`
: Specifies the non-inverted sources of the clock.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-waveform` *`float...`* Specifies the rise and fall edge times of the clock waveform over one clock period. The first value corresponds to the first rising transition after time zero. The numbers should represent one full clock period. If you omit this option, a default waveform is assumed: the leading edge occurs at 0 and the trailing edge occurs at the midpoint of the period, such that a symmetric clock is generated.

Examples

- The following command creates clock clk1 on port clkA with a period of 5 time units. The clock has a rising edge at 0 time units and a falling edge at 2.5 time units.

```
create_clock [get_ports clkA]  -name clk1  -period 5  -waveform {0 2.5}
```

- The following command specifies a comment for the to be created clock `abc`.

```
create_clock -name abc -period 1000 -comment "mycomment for clock abc"
```

Related Information

Related attribute: (clock) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

create_generated_clock

```
create_generated_clock
     -source {port|pin}
     [-add] [-master_clock clock]
     [-name clock] [-domain domain]
     [-combinational]
     [-divide_by integer]
     [-multiply_by integer [-duty_cycle float]]
     [-edges integer [integer]...
       [-edge_shift float [float]...]]
     [-invert] [-apply_inverted {port|pin}...]
     [-comment string]
     {port|pin}...
```

Creates a new clock signal from the clock waveform of a given pin in the design.

Note: This command must be specified with one of the following options: `-combinational`, `-divide_by`, `-multiply_by` or `-edges`.

Options and Arguments

`-add`

Allows to specify multiple generated clocks on the same source when multiple clocks must fan into the source pin.

If you omit this option and a clock was already defined on a pin or port, this definition would overwrite the previous one.

Note: You must specify this option together with the `-master_clock` and `-name` options.

`-apply_inverted {port|pin|hpin}`

Specifies the pin or port on which to apply the inverted clock waveform.

Note: This is non-SDC option.

`-combinational`

Restricts the path search for latency to only paths which do not cross sequential devices, or the source pins of other generated clocks.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-comment` *`string`*
: Specifies a comment to be tagged to this command.

`-divide_by` *`integer`*
: Determines the frequency of the new clock by dividing the frequency of the source clock by this factor.

  If the frequency is divided by a certain factor, the clock period is multiplied by that same factor.

  If the factor is a power of 2 (2,4,8... except 1), the rising edges of the master clock are used to determine the edges of the generated clock, otherwise the edges are scaled from the master clock edges.

`-domain` *`domain`*
: Specifies the clock domain to which the generated clock belongs.

  Default: `domain_1`

  Note: This is non-SDC option.

`-duty_cycle` *`float`*
: Specifies the duty cycle (high pulse width) as a percentage of the clock period. Specify a number between 0 and 100.

`-edge_shift` *`float`* `[`*`float`*`]...`
: Specifies how much each edge specified with the `-edges` option should be shifted.

`-edges` *`integer`* `[`*`integer`*`]...`
: Selects a list of edges from the source clock that form the edges of the derived clock.

  You must specify an odd number of edges. The last edge represents the first edge of the next clock period.

  Note: You cannot specify this option with either the `-divide_by` or `-multiply_by` option.

`-invert`
: Inverts the resulting waveform of the generated clock.

`-master_clock` *`clock`*
: Derives the generated (target) clock from the specified clock.

  Use this option if multiple signals arrive at the source pin and you want to indicate a particular clock from which the target clock must be generated.

  Note: You must specify this option together with the `-add` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-multiply_by` *`integer`*

Determines the frequency of the new clock by multiplying the frequency of the source clock with this factor.

If the frequency is multiplied with a certain factor, the clock period is divided by that same factor.

`-name` *`clock`*

Specifies the name of the generated clock.

Note: You must specify this option with the `-add` option to ensure that clocks with the same source have unique names.

{*`port`*|*`pin`*|*`hpin`*}

Specifies the pins and ports to which the generated clock must be applied

`-source` {*`port`* | *`pin`* | *`hpin`*}

Specifies the name of the pin from which the clock must be derived.

Examples

- The following command derives target clock `gen_clk1` from the clock at pin `CK` on instance `inst1` and divides the frequency by 2. The clock is applied to pin `Q` of instance `inst1`. A comment is tagged to the command.

```
create_generated_clock -name gen_clk1 -source [get_pins inst1/CK] \
-divide_by 2 [get_pins inst1/Q] -comment "mycomment for clock gen_clk1"
```

- The following command derives target clock `gen_clk1` from the clock at pin `CK` on instance `inst1` by specifying edges 1, 5 and 7. This implies a `divide_by 3` and a duty cycle of 66.66%.

```
create_generated_clock name gen_clk1 source [get_pins inst1/CK]
-edges {1 5 7} [get_pins inst1/Q]
```

- The following command shows how to create a clock by using edge numbers of the source waveform to specify the new waveform, as shown in Figure 19-1,:

```
create_generated_clock -name genclk -source pinX -edges {1 2 4 5 6} pinY
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Figure 19-1 Derived Clock (Frequency Division)

Related Information

Related attributes: (clock) comment
divide_waveform
waveform

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

current_design

```
current_design
     [design]
```

Changes the current directory in the design hierarchy to the specified design and returns the path to the design.

If no design is specified, the command returns the top-level design.

In some cases multiple designs may be in memory, either from `derive_environment` or from reading and elaborating more than a single design's RTL. In this case an error will be issued if no single design is specified.

Options and Arguments

design	Specifies the name of the top-level design.

Examples

- The following command changes the current directory in the design hierarchy to the specified design `top`.

```
legacy_genus:/> current_design top
/designs/top
legacy_genus:/designs/top>
```

- Assume the top design is called `test`. The following command returns the path to the top-level design.

```
legacy_genus:/> current_design
/designs/test
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

current_design_name

```
current_design_name
```

If the current directory is the top-level design, this command returns the design name.

If the current directory is a hierarchical instance, this commands returns its corresponding subdesign.

In all other cases, this command returns null.

Examples

- The following command returns the design name.

```
legacy_genus:/designs/topmost> current_design_name
/designs/topmost
```

- The following command returns the name of the subdesign that corresponds to hierarchical instance `inst_top`.

```
legacy_genus:/designs/topmost/instances_hier/inst_top> current_design_name
/designs/topmost/subdesigns/top
```

- The following command returns null because the instance `inst2` is a leaf-level instance.

```
legacy_genus:/designs/topmost/instances_hier/inst_top/instances_comb/inst2>
current_design_name
legacy_genus:/designs/topmost/instances_hier/inst_top/instances_comb/inst2>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

current_instance

```
current_instance
     [instance]
```

Changes the current directory in the design hierarchy to the specified instance and returns the path to the instance.

Current instances can be set to a hierarchical instance as well as to a leaf instance.

If no instance is specified, the command returns the path to the top-level design.

Options and Arguments

instance	Specifies the name of the instance to become the current instance. If the instance is not given, the current instance is set to the top instance of the hierarchy

Examples

- The following command changes the current directory in the design hierarchy to the specified instance.

```
legacy_genus:/> current_instance instA
/designs/top/instances_seq/instA
legacy_genus:/designs/top/instances_seq/instA>
```

- The following command returns the path to the top-level design.

```
legacy_genus:/designs/top/instances_seq> current_instance
/designs/top
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

filter_collection

```
filter_collection
     [-quiet] [-regexp [-nocase]]
     string...
     filter_expression
```

Returns a collection of objects that match the specified filter expression.

Note: The filter expression will be evaluated on all instances, pins, and ports in the design if the object list is empty.

Options and Arguments

filter_expression	Corresponds to a filter expression supported for the specified objects. **Note:** Genus will generate an error message if the returned collection is empty.
-nocase	Indicates that the specified pattern is not case sensitive. **Note:** This option can only be used with the -regexp option.
-regexp	Allows to specify regular expressions in the filter expressions. If this option is not specified, the filter expression can only contain wild cards (* and ?).
string	Specifies the collection of objects from which to filter objects for the new collection. The string can have any of the following types:
-quiet	Suppresses all the generated messages.

Examples

- The following command returns all instances whose libcell matches inv1.

```
legacy_genus:/> filter_collection [get_cells *] "ref_name==inv1"
/designs/top_most/instances_comb/inst2 /designs/top_most/instances_comb/inst5
```

- The following command searches all instances, pins and ports in the design for an object whose name starts with abc.

```
legacy_genus:/> filter_collection "" "base_name==abc"
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command returns all instances whose name start with either `CPF` or `bw_reg`. The case of the specified strings must be matched.

```
legacy_genus:/> filter_collection -regexp [find / -inst *] name=~"(CPF|
bw_reg).*"
/designs/chip_top/instances_hier/CPF_ISO_HIER_INST_305 /designs/chip_top/
instances_hier/bw_reg_1
```

- The following command returns all instances whose name start with either the `cpf` or `BW_REG` string without necessarily matching the case of the specified strings.

```
legacy_genus:/> filter_collection -regexp -nocase [find / -inst *]\
name=~"(CPF|bw_reg).*"
/designs/chip_top/instances_hier/CPF_ISO_HIER_INST_305 /designs/chip_top/
instances_hier/bw_reg_1
```

Related Information

For a list of supported filter expressions, refer to:

- get_cells on page 1424
- get_pins on page 1454
- get_ports on page 1458

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

foreach_in_collection

```
foreach_in_collection
     iter object_list body
```

Executes a foreach loop over the elements of a collection.

Options and Arguments

body	Specifies the body of the foreach loop.
iter	Specifies the iteration variable.
object_list	Specifies the collection over which to iterate.

Example

Consider the following collection:

```
legacy_genus:/> set list_temp [get_cells *]
/designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2
```

The following command prints for each instance in the collection the libcell:

```
legacy_genus:/> foreach_in_collection icell $list_temp \
==> {puts "cell = [get_attribute libcell $icell]"}
```

```
cell = /libraries/lca300kv/libcells/FD1
cell = /libraries/lca300kv/libcells/BUF8A
cell = /libraries/lca300kv/libcells/BUF8A
cell = /libraries/lca300kv/libcells/MUX21S
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_arcs

```
get_arcs [-filter string]
     [-of_objects instance | net ]
     [-to pin|port...] [-from pin|port...]
```

Returns a collection of arcs. Using the command options you can filter the elements that are returned.

Note: This command was added in SDC 1.5.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the instance.

`-filter "from_pin{==|=~|!=|!~}pin_name"`

`from_pin` checks the name of the from pin of the arc.

`-filter "obj_type{==|=~|!=|!~}libarc"`

`obj_type` checks if the object is a timing arc.

`-filter "sdf_cond{==|=~|!=|!~}sdf_cond"`

`sdf_cond` checks the SDF condition of the arc.

`-filter "timing_type{==|=~|!=|!~}timing_type"`

`timing_type` checks the timing type (positive_unate, negative_unate, non_unate) of the arc.

`-filter "when{==|=~|!=|!~}when_cond"`

`when` checks the when condition of the arc.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This is a non-SDC option.

`-of_objects` *`instance`* | *`net`*

Specifies to return the arcs for the specified instance or connected to the specified nets (evaluated by `get_nets` *`pattern`*).

`-from` *`pin`* | `port` Returns all the outgoing arcs of the list of pins provided.

`-to` *`pin`* | `port` Returns all the incoming arcs of the list of pins provided.

Examples

- Sample usage:

```
legacy_genus:/> get_arcs -from [get_pins *] -filter { (is_cellarc == false ) \
&& (is_disabled == true ) }
```

```
legacy_genus:/> get_arcs -from [get_pins flop1/CK] \
-filter {when == "(((!(R))&(!(SE)))&(!(SI)))"}
```

- Sample usage using `filter_collection`:

```
legacy_genus:/> filter_collection [get_arcs -from [get_pins *] ] \
{delay_max_rise < 20}
```

```
legacy_genus:/> filter_collection [get_arcs -from [get_pins *]] {is_disabled
== true && is_cellarc == true}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_cells

```
get_cells [-filter string]
     [ -hierarchical | -of_objects list [-leaf]
     | pattern [-hsc string] [-regexp [-nocase]] ]
     [-quiet]
```

Returns the full path to a list of instances. Using the command options you can filter the elements that are returned.

Note: This command was added in SDC 1.5.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "area{==|!=|<=|>=|<|>}float"`

`area` checks the area of the instance.

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the instance.

`-filter "{full_name|hierarchical_name}{==|=~|!=|!~}vname"`

`full_name` or `hierarchical_name` checks the vdir name of the instance.

`-filter "is_combinational{==|!=}{true|false}"`

`is_combinational` checks if the instance is combinational.

`-filter "is_dont_touch{==|!=}{true|false}"`

`is_dont_touch` checks if the instance is preserved.

`-filter "is_hierarchical{==|!=}{true|false}"`

`is_hierarchical` checks if the instance is hierarchical.

`-filter "is_icg{==|!=}{true|false}"`

`is_icg` checks if the cell is an integrated clock-gating cell.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-filter "is_sequential{==|!=}{true|false}"`

`is_sequential` checks if the instance is sequential.

`-filter "object_type{==|=~|!=|!~}instance"`

`object_type` checks if the object is an instance.

`-filter "pin_count{==|!=|<=|>=|<|>}integer"`

`pin_count` checks the pin count of the instance.

`-filter "ref_libcell_name{==|=~|!=|!~}libcell"`

`ref_libcell_name` checks the libcell name of the instance.

`-filter "ref_name{==|=~|!=|!~}libcell"`

`ref_name` checks the libcell name of the instance.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

`-hierarchical`

Selects all instances in the hierarchy below the current level that match the specified pattern.

`-hsc` *`string`*

Specifies the hierarchical delimiter for patterns.

For example, if the hierarchical delimiter is `@`, the `sub/I1@I2` pattern matches instance `I2` within the hierarchy `sub/I1`.

`-leaf`

Returns all leaf cells found for the *nets* specified with the `-of_objects` option after traversing the hierarchical boundaries.

Without the `-leaf` option, the command returns all hierarchical and leaf cells found at the same level of the nets specified with the `-of_objects` option without traversing the hierarchical boundaries

Note: This option can only be specified with the `-of_objects` option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option. If `-regexp` is omitted, the pattern is assumed to be the wildcard.
`-of_objects` *`list`*	Specifies to return the instances with the specified pins (evaluated by `get_pins` *`pattern)`* or connected to the specified nets (evaluated by `get_nets` *`pattern`*).
`pattern`	Specifies a pattern of cell names or regular expressions. *Default*: *
`-quiet`	Suppresses all the generated messages.
`-regexp`	Specifies that the pattern is a regular expression.

Examples

- The following command returns those instances which were mapped to libcell `flopped`.

```
legacy_genus:/> get_cells -filter "ref_name==fflopd"

/designs/top/instances_seq/inst1 /designs/top/instances_seq/instf
```

- The following command returns those instances that are named `inst1`.

```
legacy_genus:/> get_cells -filter "full_name==inst1"

/designs/top/instances_seq/inst1
```

- The following command returns those instances connected to the net `clk`.

```
legacy_genus:/> get_cells -of_objects /designs/top/nets/clk
/designs/top/instances_seq/inst3
```

- The following command returns all instances that match the regular expression pattern ^inst1.*$

```
legacy_genus:/> get_cells -regexp ^inst1.*$
/designs/top/instances_seq/inst1 /designs/top/instances_comb/inst1_0
/designs/top/instances_comb/inst1_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command uses a complex expression to return the path to
  - the list of all instances that either have libcell name inv1 or instance name `inst_mux`,
  - the list of instances that either have libcell `MX2X1` or `fflopd`

```
legacy_genus:/> get_cells -filter "((ref_name ==inv1 || full_name==inst_mux)
&& (ref_name==MX2X1)) || (ref_name==fflopd))
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_clocks

```
get_clocks
     [-filter string]
     [-regexp] [-nocase] [-quiet] [pattern]
```

Returns the full path to a list of timing clocks that match the pattern. This command reports all clocks, including inactive clocks.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the clock.

`-filter "full_name{==|=~|!=|!~}vname"`

`full_name` checks the vdir name of the clock

`-filter "has_detailed_parasitics{==|!=}{true|false}"`

`has_detailed_parasitics` checks if the `phys_capacitance` attribute is set on the clock.

`-filter "object_type{==|=~|!=|!~}clock"`

`object_type` checks if the object is a clock.

`-filter "period{==|!=|<=|>=|<|>}period"`

`period` checks the period of the clock.

`-filter "sources{==|=~|!=|!~}clock_sources"`

`sources` checks the inverted and non-inverted sources of the clock

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify -regexp in order to use this parameter. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`pattern`	Specifies a pattern of clock names. Patterns include the * and ? wildcard characters, and collections of instances and clocks. *Default*: *
`-quiet`	Suppresses all the generated messages that are generated when the `get_clocks` command is run.
`-regexpr`	Treats the specified patterns as a regular expression patterns. *Default*: Treats the specified pattern as a wild card. **Note:** Genus ignores this option for optimization and reporting purposes. The `write_sdc` command will write the option out for consumption by downstream tools.

Examples

- The following command returns the path to clock `test_clock`.

```
legacy_genus:/> get_clocks test_clock
/designs/top_most/timing/clock_domains/domain_1/test_clock
```

- The following command returns the path to all clocks starting with `test`:

```
legacy_genus:/> get_clocks test*
/designs/top_most/timing/clock_domains/domain_1/test_clock
```

- The following command returns the path to a clock whose name is `c1`.

```
legacy_genus:/> get_clocks -filter "full_name==c1"
/designs/top/timing/clock_domains/domain_1/c1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_designs

```
get_designs
     [-filter string] [-quiet] pattern
```

Returns the full path to any design or subdesign matching the pattern.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the name of the design.

`-filter "full_name{==|=~|!=|!~}vname"`

`full_name` checks the vdir name of the design

`-filter "is_dont_touch{==|!=}{true|false}"`

`is_dont_touch` checks if the design is preserved.

`-filter "max_capacitance{==|!=|<=|>=|<|>}float"`

`max_capacitance` checks the `max_capacitance` attribute set on the design

`-filter "max_fanout{==|!=|<=|>=|<|>}integer"`

`max_fanout` checks the `max_fanout` attribute set on the design

`-filter "max_transition{==|!=|<=|>=|<|>}float"`

`max_transition` checks the `max_transition` attribute set on the design

`-filter "object_type{==|=~|!=|!~}design"`

`object_type` checks if the object is a design.

Note: Any filter option can be specified as `option` or `@option` in the expression.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out. **Note:** This is a non-SDC option.
`pattern`	Specifies a pattern of design or subdesign names. **Note:** The command only looks for subdesigns if no design with the specified pattern is found.
`-quiet`	Suppresses all the generated messages.

Examples

- The following command returns the path to design `top_most`.

```
legacy_genus:/> get_designs top_most*
/designs/top_most
```

- The following command returns the path to subdesign `top_most_1`:

```
legacy_genus:/> get_designs top_most_*
/designs/top_most/subdesigns/top_most_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_generated_clocks

```
get_generated_clocks
     [-quiet] [-regexp] [-nocase] [-exact]
     [-filter expr] [pattern]
```

Returns the full path to all the generated clocks that match the pattern. Using the command options you can filter the clocks that are returned.

Options and Arguments

`-filter expr`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the generated clock.

`-filter "full_name{==|=~|!=|!~}vname"`

`full_name` checks the vdir name of the generated clock

`-filter "object_type{==|=~|!=|!~}clock"`

`object_type` checks if the object is a clock.

`-filter "period{==|!=|<=|>=|<|>}period"`

`period` checks the period of the generated clock.

`-filter "sources{==|=~|!=|!~}clock_sources"`

`sources` checks the inverted and non-inverted sources of the generated clock

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

`-exact`

Performs exact pattern matching against the object name, instead of wild card or regular expression matching. Use this parameter when the object name includes special characters such as * , ? , and + .

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Note: If you specify this parameter, you cannot specify -regexp.
`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option. If `-regexp` is omitted, the pattern is assumed to be the wildcard.
`pattern`	Specifies the pattern for generated clock names to be matched. If no pattern is specified, all generated clocks are considered. Patterns include the * and ? wildcard characters, and collections of generated clocks.
`-quiet`	Suppresses all the generated messages.
`-regexp`	Treats the specified pattern as a regular expression.y *`Default`*: Treats the specified pattern as a wild card. **Note:** If you specify this parameter, you cannot specify -exact

Examples

Assume you defined the following clocks:

```
define_clock -name clk -period 10000 -domain domain_1 \
    [find /des*/*/ports_in -port Clk]
```

```
define_clock -name clk1 -period 20000 -domain domain_2 \
[find /des*/*/ports_in -port Clk1]
```

Also assume you read in the following SDC file:

```
create_generated_clock -name gen_clk -source [find /des*/*/ports_in -port Clk]
-divide_by 2 [find /des*/*/*seq/f6/ -pin CK]
```

```
create_generated_clock -name gen_clk1 -source [find /des*/*/ports_in -port Clk1]
-divide_by 2 [find /des*/*/*seq/f12/ -pin CK]
```

```
create_generated_clock -name gen_clk2 -source [find /des*/*/ports_in -port Clk1]
-divide_by 2 [find /des*/*/*seq/f13/ -pin CK]
```

- The following command returns the full path to all generated clocks:

  ```
  legacy_genus:/> get_generated_clocks
  ```

  ```
  /designs/test/timing/clock_domains/domain_1/gen_clk /designs/test/timing/
  clock_domains/domain_1/gen_clk1 /designs/test/timing/clock_domains/domain_1/
  gen_clk2
  ```

- The following command returns the path to the generated clock with period 20000.

  ```
  legacy_genus:/> get_generated_clocks -filter "period==20000.0"
  /designs/test/timing/clock_domains/domain_1/gen_clk
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command returns the path to the generated clocks whose name is not `gen_clk1`.

```
legacy_genus:/> get_generated_clocks -filter "name!=gen_clk1"
/designs/test/timing/clock_domains/domain_1/gen_clk /designs/test/timing/
clock_domains/domain_1/gen_clk2
```

- The following command returns the path to the generated clocks whose name matches `gen_clk1` or `gen_clk2`.

```
legacy_genus:/> get_generated_clocks -regexp -nocase GEN_CLK1|GEN_CLK2
/designs/test/timing/clock_domains/domain_1/gen_clk1 /designs/test/timing/
clock_domains/domain_1/gen_clk2
```

Related Information

Related command: create_generated_clock on page 1412

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_leaf_cells

```
get_leaf_cells
     pattern
```

Returns the full path to all the leaf cells that match the pattern.

Options and Arguments

pattern	Specifies the pattern for leaf cells to be matched.

Example

- The following example matches sss*g1 cells:

```
legacy_genus:\> get_leaf_cells sss*g1
/design/test/sss/inst/g1
/design/test/sss/inst/sssg1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_leaf_nets

```
get_leaf_nets
     pattern
```

Returns the full path to all the leaf nets that match the pattern.

Options and Arguments

pattern	Specifies the pattern for leaf nets to be matched.

Example

`get_leaf_nets sss*` can match `hnet:test/sss/y` and `hnet:test/sss/a`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_leaf_pins

```
get_leaf_pins
     pattern
```

Returns the full path to all the leaf pins that match the pattern.

Options and Arguments

pattern	Specifies the pattern for leaf pins to be matched.

Example

- The following example matches sss*g1 pins:

```
legacy_genus:\> get_leaf_pins sss*g1
/design/test/inst/sss/g1/pin/z
/design/test/inst/sss/g1/pin/in_0
/design/test/inst/sssg1/z
/design/test/inst/sssg1/in_0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_libs

```
get_libs
     [-filter string]
     [-regexp] [-nocase] [-quiet]
     {-of_objects object_list | pattern_list }
```

Returns the full path to any library that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the library.

`-filter "object_type{==|=~|!=|!~}library"`

`object_type` checks if the object is a library.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

`-nocase`

Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option.

If `-regexp` is omitted, the pattern is assumed to be the wildcard.

`-of_objects object_list`

Creates a collection of libraries that belong to the library cells in the given object list.

pattern

Specifies a pattern of library names or regular expressions.

`-quiet`

Suppresses all the generated warning and error messages.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-regexp`	Specifies that the pattern is a regular expression.

Examples

- The following command just specifies a pattern to find the path to the `tutorial` library.

```
legacy_genus:/> get_libs tut*
/libraries/tutorial
```

- The following command specifies a pattern with regular expression to find the path to the `tutorial` library.

```
legacy_genus:/> get_libs -regexp tut.*
/libraries/tutorial
```

- The following command specifies a pattern that is case insensitive to find the path to the `tutorial` library.

```
legacy_genus:/> get_libs -regexp -nocase TUT.*
/libraries/tutorial
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_lib_arcs

```
get_lib_arcs
     [-to libpin...] [-from libpin...]
     [-of_objects libcell...] [-filter string]
```

Returns all timing arcs that match the specified pattern.Assign these library timing arcs to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the name of the timing arc.

`-filter "from_lib_pin{==|=~|!=|!~}pin_name"`

`from_lib_pin` checks the name of the from pin of the arc.

`-filter "object_type{==|=~|!=|!~}libarc"`

`object_type` checks if the object is a timing arc.

`-filter "sdf_cond{==|=~|!=|!~}sdf_cond"`

`sdf_cond` checks the SDF condition of the arc.

`-filter "timing_type{==|=~|!=|!~}timing_type"`

`timing_type` checks the timing type (positive_unate, negative_unate, non_unate) of the arc.

`-filter "when{==|=~|!=|!~}when_cond"`

`when` checks the when condition of the arc.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-from` *`lib_pins`*

Returns all outgoing timing arcs for the specified library cell pin(s). You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection.If you specify this parameter with the `-to` parameter, the software creates a collection of all library arcs that exist between the `to` and `from` library pins.

Note: You cannot specify this parameter with the `-of_objects` parameter.

`-of_objects` *`cells`*

Returns all timing arcs for the specified library cells.

You can specify a list of library cells or a list of timing arcs, but you cannot specify a list that contains both library cells and timing arcs. You can specify a pattern for the library cell or timing arc name, a collection of library cells or timing arcs, or a pattern and a collection.

Note: You cannot specify this parameter with the `-to` or `-from` parameter.

`-to` *`lib_pins`*

Returns all incoming timing arcs for the specified library cell pin(s).

You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection.

If you specify this parameter with the `-from` parameter, the software creates a collection of all library arcs that exist between the `to` and `from` library pins.

Note: You cannot specify this parameter with the `-of_objects` parameter.

Examples

- The following command returns the timing arcs to pin `Y` of cells `nand2` and `nor2`.

```
legacy_genus:/> get_lib_arcs -to "nand2/Y nor2/Y"
/libraries/tutorial/libcells/nand2/Y/inarcs/A_Y_n60 /libraries/tutorial/
libcells/nand2/Y/inarcs/B_Y_n60 /libraries/tutorial/libcells/nor2/Y/inarcs/
A_Y_n60 /libraries/tutorial/libcells/nor2/Y/inarcs/B_Y_n60
```

- The following command returns the timing arcs between pins `A` and `Y` of library cell `nand2`.

```
legacy_genus:/> get_lib_arcs -to nand2/Y -from nand2/A
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
/libraries/tutorial/libcells/nand2/Y/inarcs/A_Y_n60
```

- The following command returns all timing arcs for libcell `fflopd_ckn`.

```
get_lib_arcs -of fflopd_ckn
```

```
/libraries/tutorial/libcells/fflopd_ckn/D/inarcs/CKN_D_S50 /libraries/
tutorial/libcells/fflopd_ckn/D/inarcs/CKN_D_H50 /libraries/tutorial/libcells/
fflopd_ckn/Q/inarcs/CKN_Q_c50
```

- The following command filters based on the value of the `timing_type`.

```
legacy_genus:/> get_lib_arcs -from nand2/A -to nand2/Y \
-filter "timing_type==negative_unate"
```

```
/libraries/tutorial/libcells/nand2/Y/inarcs/A_Y_n60
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_lib_cells

```
get_lib_cells
     [-filter string] [-of_objects string] [-hsc string]
     [-regexp] [-nocase] [-quiet] [pattern]
```

Returns the full path to any library cell that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "area{==|!=|<=|>=|<|>}float"`

`area` checks the area of the libcell

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the libcell.

`-filter "is_combinational{==|!=}{true|false}"`

`is_combinational` checks if the libcell is combinational

`-filter "is_dont_touch{==|!=}{true|false}"`

`is_dont_touch` checks if the libcell is preserved.

`-filter "is_dont_use{==|!=}{true|false}"`

`is_dont_use` checks if the libcell must be avoided.

`-filter "is_sequential{==|!=}{true|false}"`

`is_sequential` checks if the libcell is sequential.

`-filter "is_tristate{==|!=}{true|false}"`

`is_tristate` checks if the libcell is tristate.

`-filter "object_type{==|=~|!=|!~}libcell"`

`object_type` checks if the object is a libcell.

Note: Any filter option can be specified as `option` or `@option` in the expression.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out. **Note:** This is a non-SDC option.
`-hsc` *`string`*	Specifies the hierarchical delimiter for patterns.
`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option. If `-regexp` is omitted, the pattern is assumed to be the wildcard.
`-of_objects` *`string`*	Specifies the libpins or instances for which to return the libcells.
`pattern`	Specifies a pattern of libcell names or regular expressions.
`-quiet`	Suppresses all the generated messages.
`-regexp`	Specifies that the pattern is a regular expression.

Examples

- The following command returns all libcells whose name start with `nan`.

```
legacy_genus:/> get_lib_cells nan*
/libraries/tutorial/libcells/nand2
```

- The following command returns all libcells whose name start with `nan` and that are part of the library whose name starts with `tut`.

```
legacy_genus:/> get_lib_cells tut*/nan*
/libraries/tutorial/libcells/nand2
```

- The following command returns all libcells whose name start with `XOR3XL`.

```
legacy_genus:/> get_lib_cells XOR3XL*
/libraries/library_domains/set1/typical/libcells/XOR3XL
/libraries/library_domains/set2/typical/libcells/XOR3XL
```

  In this case, two library sets were read in which each contained the `typical` library with their version of the `XOR3XL` cell.

- The following command returns all libcells whose name start with `XOR3XL` and that belong to the typical library.

```
legacy_genus:/> get_lib_cells typical@XOR3XL -hsc @
/libraries/library_domains/set1/typical/libcells/XOR3XL
/libraries/library_domains/set2/typical/libcells/XOR3XL
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_lib_pins

```
get_lib_pins
     [-filter string] [-of_objects string] [-hsc string]
     [-regexp] [-nocase] [-quiet] [pattern]
```

Returns the full path to any libcell pin that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the libcell pin.

`-filter "capacitance{==|!=|<=|>=|<|>}float"`

`capacitance` checks the average capacitance (rise+fall/2) of the libcell pin.

`-filter "direction={in|out|inout}"`

`direction` checks the direction of the libcell pin.

`-filter "is_async_pin{==|!=}{true|false}"`

`is_async_pin` checks if the libcell pin is an asynchronous pin.

`-filter "is_clear_pin{==|!=}{true|false}"`

`is_clear_pin` checks if the libcell pin is a clear pin.

`-filter "is_clock_pin{==|!=}{true|false}"`

`is_clock_pin` checks if the libcell pin is a clock pin.

`-filter "max_capacitance{==|!=|<=|>=|<|>}float"`

`max_capacitance` checks the `max_capacitance` attribute set on the libcell pin.

`-filter "max_fanout{==|!=|<=|>=|<|>}integer"`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`max_fanout` checks the `max_fanout` attribute set on the libcell pin.

`-filter "max_transition{==|!=|<=|>=|<|>}float"`

`max_transition` checks the `max_transition` attribute set on the libcell pin.

`-filter "object_type{==|=~|!=|!~}libpin"`

`object_type` checks if the object is a libpin.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

`-hsc` *`string`* — Specifies the hierarchical delimiter for patterns.

`-nocase` — Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option.

If `-regexp` is omitted, the pattern is assumed to be the wildcard.

`-of_objects` *`string`*

Specifies the libcells or pins for which to return the libcell pins.

`pattern` — Specifies a pattern of libpin names or regular expressions.

`-quiet` — Suppresses all the generated messages.

`-regexp` — Specifies that the pattern is a regular expression.

Examples

In the following examples, two library sets were read in. Each library set contained the `typical` library with its version of the `XOR3XL` cell. The examples show different ways to get the same information.

- The following command returns all libpins whose name start with `A` and that belong to a libcell whose name starts with `XOR3XL`.

```
legacy_genus:/> get_lib_pins XOR3XL*/A*
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
/libraries/library_domains/set1/typical/libcells/XOR3XL/A
/libraries/library_domains/set2/typical/libcells/XOR3XL/A
```

- The following command uses the hierarchy delimiter to request to return all libpins whose name start with A and that belong to a libcell whose name starts with `XOR3XL`.

```
legacy_genus:/> get_lib_pins XOR3XL*@A* -hsc @
```

```
/libraries/library_domains/set1/typical/libcells/XOR3XL/A
/libraries/library_domains/set2/typical/libcells/XOR3XL/A
```

- The following command returns all libpins whose name start with `A` and that belong to a libcell whose name start with `xor3xl` but indicates that the pattern is not case sensitive.

```
legacy_genus:/> get_lib_pins -regexp -nocase xor3xl/A.*
```

```
/libraries/library_domains/set1/typical/libcells/XOR3XL/A
/libraries/library_domains/set2/typical/libcells/XOR3XL/A
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_nets

```
get_nets
     [-filter string]
     [-hierarchical] [-hsc string] [-regexp]
     [-nocase] [-quiet] [-of_objects string | pattern]
     [-segments [-top_net_of_hierarchical_group]]
```

Returns the full path to any net that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "area{==|!=|<=|>=|<|>}float"`

`area` checks the enclosed area of the net

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the net.

`-filter "full_name{==|=~|!=|!~}vname"`

`full_name` checks the vdir name of the net

`-filter "is_dont_touch{==|!=}{true|false}"`

`is_dont_touch` checks if the net is preserved.

`-filter "object_type{==|=~|!=|!~}net"`

`object_type` checks if the object is a net.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

`-hierarchical`

Selects all the nets in the hierarchy below the current level that match the specified pattern.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hsc` *`string`*	Specifies the hierarchical delimiter for patterns.
`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option. If `-regexp` is omitted, the pattern is assumed to be the wildcard.
`-of_objects` *`string`*	Specifies to return the nets that are connected to the specified cells (evaluated by `get_cells` *`pattern`*), pins (evaluated by `get_pins` *`pattern`*), and ports (evaluated by `get_ports` *`pattern`*).
`pattern`	Specifies a pattern of net names or regular expressions. *Default*: *
`-quiet`	Suppresses all the generated messages.
`-regexp`	Specifies that the pattern is a regular expression.
`-segments`	Returns all global segments for the given nets. Global net segments are all those physically connected segments across all hierarchical boundaries. For each net, all global segments that are part of that net are added to the resulting collection. This modifies the initial search that matches patterns or objects It is recommended to use this option with a single net.
`-top_net_of_hierarchical_group`	Keeps 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. When you use this option together with the `-segments` option, you can isolate the highest net segment of a physical net.

Examples

- The following command returns net `in` that is in the `inst9` hierarchy.

```
legacy_genus:/> get_nets -hsc @ inst9@in
/designs/top_most/instances_hier/inst9/nets/in
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command returns all nets whose name start with `c`.

  ```
  legacy_genus:/> get_nets c* -filter "object_type==net"
  ```

  ```
  /designs/aco1/nets/ck {/designs/aco1/nets/c[0]} {/designs/aco1/nets/c[1]}
  {/designs/aco1/nets/c[2]} {/designs/aco1/nets/c[3]} {/designs/aco1/nets/c[4]}
  {/designs/aco1/nets/c[5]}
  ```

- The following command returns all the segments of a net starting from port `in` driving port `A` of an inverter 2 levels below: `i1_top/i1_mid/i1_inv/A`

  ```
  get_nets -segments i1_mid/in
  ```

  Returns following segments: `in, i1_top/in, i1_mid/in`

- The following command returns the net segment closest to the top of `i1_mid/in`

  ```
  get_nets -segments -top_net_of_hierarchical_group i1_mid/in
  ```

  Returns following segment: `in`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_object_name

```
get_object_name
     {library|libcell|libpin|libarc|subdesign|design|instance|port|pin
     |net|clock|exception}...
```

This command was added for compatibility while reading in SDC files with non-SDC constraints. Generally, this command is used with the command get_* in SDC. By default, get_* commands returns a pointer of the objects like get_object_name [get_* *]returns object-list of the pointer. This command can also be used to get the object name without Genus hierarchal.

Note: Since Genus does not have a concept of collections, this command just returns the specified argument. If the object name include `[]`, `get_object_name` returned value is "`{$the_object}`".

Options and Arguments

{library|libcell|libpin|libarc|design|instance|port|pin|net|clock|exception}

Specifies the argument to be returned.

Example

```
legacy_genus:/> get_object_name /designs/top/instances_hier/abc
/designs/top/instances_hier/abc
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_path_groups

```
get_path_groups
     [-quiet] [-regexp [-nocase]]
     [-filter string] [pattern]
```

Returns all the path groups that match the pattern. Using the command options you can filter the path groups that are returned.

Options and Arguments

`-filter` *`string`*	Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:
	`-filter "{base_name|name}{==|=~|!=|!~}name"`
	`base_name` or `name` checks the base name of the path group.
	`-filter "object_type{==|=~|!=|!~}path_group"`
	`object_type` checks if the object is an path_group.
	Note: Any filter option can be specified as `option` or `@option` in the expression.
	You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.
	Note: This is a non-SDC option.

`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option. If `-regexp` is omitted, the pattern is assumed to be the wildcard.
`pattern`	Specifies the pattern for path group names to be matched. If no pattern is specified all path groups are considered.
`-quiet`	Suppresses all the generated messages.
`-regexp`	Specifies that the specified pattern is a regular expression.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Examples

- Assuming you had three generated clocks, the tool will implicitly associate path groups with these clocks. The following command returns the full path to the path groups associated with the generated clocks:

```
legacy_genus:/designs/test/> get_path_groups
/designs/test/timing/exceptions/path_groups/gen_clk /designs/test/timing/
exceptions/path_groups/gen_clk1 /designs/test/timing/exceptions/path_groups/
gen_clk2
```

- The following command returns the path to a path group whose pattern matches `gen_clk1` or `gen_clk2`.

```
legacy_genus:/designs/test/> get_path_groups -regexp -nocase gen_clk1|gen_clk2
/designs/test/timing/exceptions/path_groups/gen_clk1 /designs/test/timing/
exceptions/path_groups/gen_clk2
```

- The following command returns the path to a path group whose name matches `gen_clk`.

```
legacy_genus:/designs/test/> get_path_groups -filter "name==gen_clk"
/designs/test/timing/exceptions/path_groups/gen_clk
```

Related Information

Related commands: create_generated_clock on page 1412
path_group on page 485

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_pins

```
get_pins
     [-filter string] [-hierarchical] [-hsc string] [-regexp]
     [-nocase] [-quiet] {-of_objects string [-leaf]| pattern}
```

Returns the full path to any pin that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

```
-filter "{base_name|pin_name|name}{==|=~|!=|!~}name"
```

`base_name`, `pin_name` or `name` checks the base name of the pin.

Note: If the pin name specified in the SDC file cannot be found in the design, Genus uses the patterns listed in the following mapping table to search for the pins:

Pin name in SDC	Pattern searched for in design
D, next_state, data_in	d
Q	q
CLK, CP, clocked,_on, enable	clk, ena
SEN, synch_enable	sena
SCLR, SPRE, synch_preset, synch_clear, synch_toggle	srl, srd
APRE, preset	apre
ACLR, clear	aclr

```
-filter "{direction|pin_direction}={in|out|inout}"
```

`direction` or `pin_direction` checks the direction of the pin.

```
-filter "{full_name|hierarchical_name}{==|=~|!=|
!~}vname"
```

`full_name` or `hierarchical_name` checks the vdir name of the pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-filter "fanin{==|=~|!=|!~}integer"`

`fanin` checks the fanin of the pin.

`-filter "fanout{==|=~|!=|!~}integer"`

`fanout` checks the fanout of the pin.

`-filter "is_clock_gating_pin{==|!=}{true|false}"`

`is_clock_gating_ pin` checks if the libcell pin is a clock or enable pin of a clock_gating cell.

`-filter "is_clock_pin{==|!=}{true|false}"`

`is_clock_pin` checks if the libcell pin is a clock pin.

`-filter "is_hierarchical{==|!=}{true|false}"`

`is_hierarchical` checks if the pin is hierarchical.

`-filter "is_on_clock_network{==|!=}{true|false}"`

`is_on_clock_network` checks if the libcell pin if the pin is on a clock network.

`-filter "is_pin{==|!=}{true|false}"`

`is_pin` checks if the object is a pin.

`-filter "is_port{==|!=}{true|false}"`

`is_port` checks if the object is a port.

`-filter "lib_pin_name{==|=~|!=|!~}string"`

`lib_pin_name` checks if the object is a libpin.

`-filter "object_type{==|=~|!=|!~}string"`

`object_type` checks if the object is a pin.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

Note: This is a non-SDC option.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-hierarchical`	Selects all pins in the hierarchy below the current level that match the specified pattern.
`-hsc` *`string`*	Specifies the hierarchical delimiter for patterns.
`-leaf`	Returns all leaf pins found for the *nets* specified with the `-of_objects` option after traversing the hierarchical boundaries. Without the `-leaf` option, the command returns all hierarchical and leaf pins found at the same level of the nets specified with the `-of_objects` option without traversing the hierarchical boundaries. **Note:** This option can only be specified with the `-of_objects` option.
`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option.
`-of_objects` *`string`*	Specifies to return the pins of the specified instances (evaluated by `get_cells` *`pattern`*) and nets (evaluated by `get_nets` *`pattern`*).
`pattern`	Specifies a pattern of cell names or regular expressions.
`-quiet`	Suppresses all the generated messages.
`-regexp`	Specifies that the pattern is a regular expression.

Examples

- The following command returns the full path to all the pins of instance `g6`.

```
legacy_genus:/designs/rct> get_pins -hsc @ g6@*
/designs/rct/instances_comb/g6/pins_in/A /designs/rct/instances_comb/g6/
pins_in/B /designs/rct/instances_comb/g6/pins_out/Y
```

- The following command returns the full path to pin `Y` that is part of instance `inst1`, which in turn belongs to the hierarchy of instance `inst9`.

```
legacy_genus:/> get_pins -hsc @ inst9@inst1@Y
/designs/top_most/instances_hier/inst9/instances_comb/inst1/pins_out/Y
```

- The following command returns the full path to the output pin of instance `g6`.

```
legacy_genus:/designs/rct> get_pins -hsc @ g6@* -filter direction==out
/designs/rct/instances_comb/g6/pins_out/Y
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command returns the full path to all output pins of the instances whose name start with `g`.

```
legacy_genus:/designs/rct> get_pins -filter direction==out -of_objects g*
/designs/rct/instances_comb/g3/pins_out/Y /designs/rct/instances_comb/g5/
pins_out/Y /designs/rct/instances_comb/g4/pins_out/Y
```

- The following command returns the full path to all output pins to which nets `n_1` and `n_3` connect.

```
legacy_genus:/designs/rct> get_pins  -filter direction==out -of_objects {n_1
n_3}
/designs/rct/instances_comb/g6/pins_out/Y /designs/rct/instances_comb/
cdn_loop_breaker/pins_out/Y
```

- The following command checks if the pin inst1/A specified in the filter expression exist and then returns the path to the pin:

```
legacy_genus:/ get_pins -filter "full_name==inst1/A"
/designs/top/instances_comb/inst1/A
```

- The following commands return nothing:

```
legacy_genus:/> get_pins -filter "is_port==true" *
legacy_genus:/> get_ports -filter "is_pin==false" *
```

- The following command returns all pins connected by net `ck`.

```
legacy_genus:/> get_pins -filter "object_type==pin" -of_object /designs/aco1/
nets/ck
{/designs/aco1/instances_seq/c_reg[3]/pins_in/clk} {/designs/aco1/
instances_seq/c_reg[0]/pins_in/clk} {/designs/aco1/instances_seq/c_reg[4]/
pins_in/clk} {/designs/aco1/instances_seq/c_reg[5]/pins_in/clk} {/designs/
aco1/instances_seq/c_reg[2]/pins_in/clk}
```

- The following example shows the effect of the `-quiet` option.

```
legacy_genus:/> get_pins invalid_pin
Warning : Could not find requested search value. [SDC-208] [get_pins]
        : The 'get_pins' command  cannot find any pins named 'invalid_pin'
        : Use the 'cd' and 'ls' commands to browse the virtual directories to
find the object because the specified name and/or location does not exist.
legacy_genus:/> get_pins -quiet invalid_pin
legacy_genus:/>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_ports

```
get_ports
     [-filter string] [-regexp] [-nocase] [-quiet]
     [-of_objects string | pattern]
```

Returns the full path to any port that matches the pattern. Using the command options you can filter the elements that are returned.

Options and Arguments

`-filter string`

Filters the result based on the specified expression evaluating to `true`. The following filter expressions are supported:

`-filter "{base_name|name}{==|=~|!=|!~}name"`

`base_name` or `name` checks the base name of the port.

`-filter "{direction|port_direction}={in|out|inout}"`

`direction` or `port_direction` checks the direction of the port.

`-filter "full_name{==|=~|!=|!~}vname_of_port"`

`full_name` checks the vdir name of the port

`-filter "is_on_clock_network{==|!=}{true|false}"`

`is_on_clock_network` checks if the libcell pin if the pin is on a clock network.

`-filter "is_pin{==|!=}{true|false}"`

`is_pin` checks if the object is a pin.

`-filter "is_port{==|!=}{true|false}"`

`is_port` checks if the object is a port.

`-filter "max_capacitance{==|!=|<=|>=|<|>}float"`

`max_capacitance` checks the maximum capacitance value set on the port

`-filter "max_fanout{==|!=|<=|>=|<|>}integer"`

`max_fanout` checks the `max_fanout` attribute set on the port

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-filter "max_transition{==|!=|<=|>=|<|>}float"`

`max_transition` checks the `max_transition` attribute set on the port

`-filter "object_type{==|=~|!=|!~}string"`

`object_type` checks if the object is a port.

Note: Any filter option can be specified as `option` or `@option` in the expression.

You can also combine two or more expressions using the following operators: `AND`, `OR`, `&&`, and `||`. You can use parentheses `()` to enclose simple filter expressions. However, if any of the individual filter expressions contains an `&`, `|`, `AND`, or `OR`, the command errors out.

`-nocase`	Indicates that the pattern is not case sensitive. This option can only be used with the `-regexp` option.
`-of_objects` *`string`*	Specifies to return the ports connected to the specified nets (evaluated by `get_nets` *`pattern`*).
`pattern`	Specifies a pattern of port names or regular expressions. *Default*: *
`-quiet`	Suppresses all the generated warning and error messages.
`-regexp`	Specifies that the pattern is a regular expression.

Examples

- The following command returns the full path to all input ports starting with letter `d`.

```
legacy_genus:/designs/rct> get_ports -filter direction==in d*
```

```
{/designs/rct/ports_in/din[15]} {/designs/rct/ports_in/din[14]} {/designs/
rct/ports_in/din[13]} {/designs/rct/ports_in/din[12]} {/designs/rct/ports_in/
din[11]} {/designs/rct/ports_in/din[10]} {/designs/rct/ports_in/din[9]} {/
designs/rct/ports_in/din[8]} {/designs/rct/ports_in/din[7]} {/designs/rct/
ports_in/din[6]} {/designs/rct/ports_in/din[5]} {/designs/rct/ports_in/
din[4]} {/designs/rct/ports_in/din[3]} {/designs/rct/ports_in/din[2]} {/
designs/rct/ports_in/din[1]} {/designs/rct/ports_in/din[0]} /designs/rct/
ports_in/done
```

- The following command returns the full path to all ports connected to nets whose name start with `rd_addr`.

```
legacy_genus:/designs/rct> get_ports -of_objects rd_addr*
```

```
{/designs/rct/ports_in/rd_addr[0]} {/designs/rct/ports_in/rd_addr[9]} {/
designs/rct/ports_in/rd_addr[11]} {/designs/rct/ports_in/rd_addr[2]} {/
designs/rct/ports_in/rd_addr[13]} {/designs/rct/ports_in/rd_addr[4]} {/
designs/rct/ports_in/rd_addr[15]} {/designs/rct/ports_in/rd_addr[6]} {/
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
designs/rct/ports_in/rd_addr[8]} {/designs/rct/ports_in/rd_addr[1]} {/
designs/rct/ports_in/rd_addr[10]} {/designs/rct/ports_in/rd_addr[12]} {/
designs/rct/ports_in/rd_addr[3]} {/designs/rct/ports_in/rd_addr[14]} {/
designs/rct/ports_in/rd_addr[5]} {/designs/rct/ports_in/rd_addr[7]}
```

- The following commands return nothing:

```
legacy_genus:/> get_ports -filter "is_pin==true" *
legacy_genus:/> get_pins -filter "is_port==false" *
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

get_references

```
get_references
     {subdesign | instance} [-quiet]
```

Returns

- The list of instances of the specified *subdesign*
- The subdesign of the specified *hierarchical instance*
- The libcell of the specified *leaf instance*

Options and Arguments

instance	Specifies the instance for which to get the reference. You can specify a leaf instance or a hierarchical instance.
subdesign	Specifies the subdesign for which to get the reference.
-quiet	Suppresses all the generated messages.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

getenv

```
getenv
     env_variable
```

Returns the contents of the specified environment variable.

Options and Arguments

env_variable	Specifies the environment variable for which you want to display the contents.

Example

The following command displays the content of the `CDN_SYNTH_ROOT` variable.

```
legacy_genus:/> getenv CDN_SYNTH_ROOT
/net/your_install_dir/tools.lnx86
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

group_path

```
group_path
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|net|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|net|port_bus|pin_bus}...
      |-fall_through {port|instance|pin|net|port_bus|pin_bus}...}...
     [-rise] [-fall]
     [-setup] [-hold]
     [-exception_name name]
     {-name cost_group | -default}
     [-weight float]
     [-critical_range float]
     [-comment string]
```

Assigns paths that meet the path selection criteria to a cost group. Paths can be selected using the `-from`, `-fall_from`, `-rise_from`, `-through`, `-fall_through`, `-rise_through`, `-to`, `-fall_to`, or `-rise_to` options. You must provide at least one of these options.

Options and Arguments

-comment string	Specifies a comment to be tagged to this command.
-critical_range	**Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
-default	Associates the path group with the default cost group
-exception_name *name*	Specifies the name of the timing exception this path group should be associated with.
-fall	Limits the path selection to those paths who have a falling edge at the end points.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall_from {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths whose signals have a falling edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-fall_through {port|inst|hinst|pin|hpin|hnet|port_bus|hpin_bus}`

Selects all paths whose signals have a falling edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-fall_to {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths whose signals have a falling edge at the specified end points.The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-from {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths that start from the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-hold`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-name cost_group` Specifies the name of the cost group to which this path group belongs.

The cost group should have been previously defined with a `define_cost_group` command.

`-rise` Limits the path selection to those paths who have a rising edge at the end points.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-rise_from {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths whose signals have a rising edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-rise_through {port|inst|hinst|pin|hpin|hnet|port_bus|hpin_bus}`

Selects all paths whose signals have a rising edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-rise_to {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths whose signals have a rising edge on the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-setup` Indicates that the path groups apply for setup analysis only.

`-through {port|inst|hinst|pin|hpin|hnet|port_bus|hpin_bus}`

Selects all paths that traverse through the specified points. The points to traverse can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-to {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-weight float` Sets a weight for the cost group with which this path group is associated.

Examples

- The following example assigns the paths from all inputs to all outputs to the cost group called `I2O`.

```
group_path -from /designs/*/ports_in/* -to /designs/*/ports_out/* \
-name I2O -comment " my path_group"
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following two commands are equivalent. They both define path groups for cost group C2C that have a rising edges at the end points.

```
group_path -rise -to inst1/pin1 -name C2C
group_path -rise_to inst1/pin1 -name C2C
```

Related Information

Related commands:	define_cost_group on page 461
	path_group on page 485
	specify_paths on page 494
Related attribute:	(exception) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

index_collection

```
index_collection
     collection integer
```

Searches an ordered list of objects and creates a new collection containing the object at the specified index.

Note: The first object in the list is at index 0.

An error will be given if the collection is empty.

Options and Arguments

collection	Specifies the name of a collection of objects to be searched.
integer	Specifies the index of the object to be returned. Valid values range from 0 till the length of the list -1. If the specified index is outside of the allowed range, an error message will be returned.

Examples

- The following example creates a new collection containing the second object of the original collection.

```
legacy_genus:/> set l2 {clk d sena aclr}
clk d sena aclr
legacy_genus:/> index_collection $l2 1
d
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

list_property

`list_property -type string`

Lists all of the properties associated with the specified object type. The supported object types are cell, pin, port, clock, libcell, libpin, lib (library), design and net.

Options and Arguments

`-type string`	Specifies the object type for which to list the properties

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

load_of

`load_of {libpin | pin}`

Returns the capacitance of the specified pin.

In case of a library pin:

- The command returns the largest of the rise and fall capacitances of the libpin.

In case of an instance pin:

- For a pin that does have a corresponding library pin, the command returns the largest of the rise and fall capacitances of the libpin.
- For a pin that has no corresponding library pin, the command returns 0.0.

Options and Arguments

libpin	Specifies the library pin for which to return the capacitance
pin	Specifies the instance pin for which to return the capacitance

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_case_analysis

`remove_case_analysis {pin|port}...`

Removes the case constant setting on the specified pins and ports.

You can set the case constant using one of the following:

- the `set_case_analysis` command
- the `timing_case_logic_value` pin or port attribute
- the `timing_case_logic_value_by_mode` pin or port attribute

Options and Arguments

pin | *port*	Removes the case constant settings from the specified pins or ports.

Related Information

Related command:	set_case_analysis on page 1531
Related attributes:	(pin) timing_case_logic_value
	(pin) timing_case_logic_value_by_mode
	(port) timing_case_logic_value
	(port) timing_case_logic_value_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_clock_gating_check

```
remove_clock_gating_check
     [-setup] [-hold]
     [-rise] [-fall]
     [-high | -low]
     [instance| pin]...
```

Removes any clock gating check value defined using the `set_clock_gating_check` command or inferred automatically by Genus.

Note: To write out this constraint in the output SDC file, you must set:

```
set write_remove_cgc 1
```

Options and Arguments

`-fall`	Removes any clock gating check specified for a falling clock transition.
`-high`	Removes any clock gating check for which the non-controlling value of the clock is high.
`-hold`	Resets clock gating hold values. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
instance | *pin*	Removes the clock gating checks from the specified instances or pins.
`-low`	Removes any clock gating check for which the non-controlling value of the clock is low.
`-rise`	Removes any clock gating check specified for a rising clock transition.
`-setup`	Removes any clock gating check that has a setup value.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_clock_latency

```
remove_clock_latency
     [-source]
     [-clock clock_list]
     {clock | port | pin}...
```

Removes the general and clock-specific latencies from the specified objects.

Note: Although supported by the `read_sdc` command, it is *not* an official SDC command.

Options and Arguments

`-clock` *`clock_list`*	Removes only the clock-specific latencies from the specified pins and ports with respect to the specified clocks. General latencies are not removed.
`clock|pin|port`	Specifies the objects (clocks, pins and ports) from which the latencies should be removed. **Note:** No clocks can be specified in the object list if the `-clock` option is specified.
`-source`	Removes the source latencies from the specified clocks, pins or ports. If this option is omitted, the network latencies are removed.

Example

The following command removes all the network latencies defined on pin `pin1` that refer to clock `clk1`.

```
remove_clock_latency -clock clk1 pin1
```

Related Information

Related command: set_clock_latency on page 1539

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_clock_transition

```
remove_clock_transition
     clock_list {-min | -max} {-rise | -fall}
```

Removes the clock transition (slew) from the specified clocks.

Options and Arguments

clock_list	Removes only the clock skews from the specified clocks.
-fall	Removes the falling transition value.
-max	Removes the maximum slew value.
-min	Removes the minimum slew value.
-rise	Removes the rising transition value.

Related Information

Related command: set_clock_transition on page 1547

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_disable_clock_gating_check

```
remove_disable_clock_gating_check
     [-setup] [-hold] [-rise] [-fall] [-high | -low]
     {pin | instance}...
```

Restores the clock gating checks inferred on the specified pins or the enable pins of the specified instances that were present before applying the `set_disable_clock_gating_check` command.

Note: This command is supported only on mapped instances and pins of mapped instances. This command is not supported on the clock pins of an instance.

Note: To write this constraint in the output SDC file, you must set the following variable:

```
set write_disable_and_rmdisable_cgc 1
```

Options and Arguments

`-fall`	Restores any clock gating check specified for a falling clock transition.
`-high`	Restores any clock gating check for which the non-controlling value of the clock is high.
`-hold`	**Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`instance`	Restores the clock gating check value(s) inferred at every single enable pin of the specified instance.
`-low`	Restores any clock gating check for which the non-controlling value of the clock is low.
`pin`	Restores the clock gating check value(s) on the specified pins.
`-rise`	Restores any clock gating check specified for a rising clock transition.
`-setup`	

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Related Information

Related command: set_disable_clock_gating_check on page 1556

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_disable_timing

```
remove_disable_timing
     { [-from pin] [-to pin] {libcell | instance}...
     | {libpin | pin | port| subport}... }
```

Restores the timing arcs through the specified objects that were disabled using the `set_disable_timing` command.

Options and Arguments

`-from` *`pin`*

Restores the timing arcs starting from this pin in the specified libcells or instances.

Note: An error is given when you specify an object type other than a libcell or instance with this option.

{*`libpin`* | *`pin`* | `port` | *`subport`*}

Specifies the libpin, pin, port, or hierarchical pin or port at which the timing path(s) must be restored.

{*`libcell`* | *`instance`*}

Specifies the libcells or instances for which you want to restore timing arcs. If you specify a libcell or instance without a `-from` or `-to` option, the tool restores all timing arcs for that libcell or instance.

`-to` *`pin`*

Restores the timing arcs ending at this pin in the specified libcells or instances.

Note: An error is given when you specify an object type other than a libcell or instance with this option.

Related Information

Related command: set_disable_timing on page 1558

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_from_collection

```
remove_from_collection
     base_collection second_collection [-intersect]
```

Removes any occurrence of an element in the second collection from the base collection and returns the new collection.

If the base and second collections have different objects, the result will be the base collection.

Note: The collections cannot contain a list of objects. If either the base or second collection or both contain a list of objects, an error message will be issued.

Options and Arguments

base_collection	Specifies the base collection.
-intersect	Provides intersection of base collection elements with second collection of objects or lists.
second_collection	Specifies the collection containing the objects that must be removed from the base collection.

Examples

- The base collection contains cells and ports. The following command indicates to remove the cells from the base collection.

```
legacy_genus:/> remove_from_collection [concat [get_cells *] [get_ports *]] \
==> [get_cells *]
```

```
/designs/test/ports_in/clk1 /designs/test/ports_in/data1 /designs/test/
ports_in/sel /designs/test/ports_out/out1
```

- The base collection contains a list. As a result, the command fails.

```
legacy_genus:/> remove_from_collection [list [get_cells *] [get_ports *]] \
==> [get_cells *]
```

```
Error   : Invalid collection. [SDC-225] [remove_from_collection]
        : The command 'remove_from_collection' does not support object list '{/
designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2}' in collection.
        : Base collection cannot contain an object list.
```

- In the following example, the objects in the base and second collection are different. Consequently, the resulting collection equals the base collection.

```
legacy_genus:/> remove_from_collection [get_cells *] [get_ports *]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
/designs/test/instances_seq/U3 /designs/test/instances_comb/U0 /designs/test/
instances_comb/U1 /designs/test/instances_comb/U2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_generated_clock

```
remove_generated_clock
     {-all | clock_list}
```

Removes all the generated clock objects created by the `create_generated_clock` command in the design.

Options and Arguments

`-all`	Removes all generated clock objects.
`clock_list`	Removes the specified list of generated clocks. You can use wildcards to specify a list of clocks.

Examples

- The following command removes the generated clock `gen_clk`.

  ```
  legacy_genus:/> remove_generated_clock [find / -clock gen_clk]
  ```

- The following command removes all generated clocks whose name start with abc.

  ```
  legacy_genus:/> remove_generated_clock [find / -clock abc*]
  ```

Related Information

Related command: create_generated_clock on page 1412

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_ideal_net

`remove_ideal_net` *`net_list`*

Causes the specified nets to be treated as non-ideal and allows the timing and optimization engines to optimize these nets. The command sets the `ideal_driver` attribute on all the leaf-level drivers of the specified nets to `false`.

The `ideal_driver` attribute can be set manually or by using the `set_ideal_net` command.

Options and Arguments

`net_list`	Specifies the nets which must be treated as non-ideal.

Related Information

Related command:	set_ideal_net on page 1580
Sets this attribute:	ideal_driver

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_ideal_network

`remove_ideal_network {pin | port | net}...`

Sets the `ideal_network` and `ideal_driver` attributes on all the leaf-level drivers of the specified net to `false`. The `ideal_network` attribute can be set manually or by the `set_ideal_network` command.

Options and Arguments

`{pin | port| net}`

Specifies the object to which the command applies.

Related Information

Related command:	set_ideal_network on page 1581
Sets these attributes:	ideal_driver
	ideal_network

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_input_delay

```
remove_input_delay
     [-clock clock] port
```

Removes the input delay on the specified port related to either the specified clock or all clocks.

Note: If an input delay is associated with several ports, it will not be removed.

Options and Arguments

`-clock` *`clock`*	Removes the input delay relative to the specified clock. If this option is omitted, the input delays relative to all clocks on this port will be removed. **Note:** This command accepts only one clock at a time. Wildcard card and lists are not supported with this switch.
`port`	Removes the input delay from the specified port. **Note:** Wild card or port lists are accepted with this switch.

Example

```
legacy_genus:/designs/rct> remove_input_delay [find / -port ena]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

remove_output_delay

```
remove_output_delay
     [-clock clock] port
```

Removes the output delay on the specified port related to either the specified clock or all clocks.

Note: If an output delay is associated with several ports, it will not be removed.

Options and Arguments

`-clock` *`clock`*	Removes the output delay relative to the specified clock. If this option is omitted, the input delays relative to all clocks on this port will be removed. **Note:** It accepts only one clock at a time, wildcards and lists are not supported with this option.
`port`	Removes the output delay from the specified port. **Note:** Wild card is accepted or a list of ports is also accepted.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report_property

```
report_property
     { collection | list_of_collections }
     -property_list property_list
```

Reports all of the specified properties associated with each object in the specified collection.

Note: For properties which return a populated collection, the command output displays {...}. For empty collections returned, a {} will be displayed.

Note: If there are different object types in the list, the property names should be common for all the objects specified in the list.

Options and Arguments

collection | list_of_collections

Specifies the collection, or list of collections for which to report.

The collection can contain any of the following object types:

library, libcell, libpin, libarc, design, subdesign, instance, port, pin, net, clock, or exception.

-property_list *property_list*

Specifies a list of valid property names.

Example

- The following command shows attributes of the specified properties:

```
report_property [ get_pins TU11/A ] -property_list {name direction}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock

```
reset_clock
     [-all] | [-clock clock_list]
```

Removes previously created clock assertions.

Options and Arguments

`-all`	Resets all clocks in the design.
`-clock` *`clock_list`*	Specifies the list of clocks for which the clock assertions are to be removed.

Examples

- The following command resets all the clock assertions from a design:

  ```
  reset_clock -all
  ```

- The following command removes the clock assertion for the clock object in a design:

  ```
  reset_clock -clock [get_clocks CK]
  ```

Related Information

Related command: report clocks on page 600

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_gating_check

```
reset_clock_gating_check
     [-setup] [-hold]
     [-rise] [-fall]
     [-high | -low]
     [inst | hinst | pin | hpin]...
```

Resets the specified setup and hold values for clock gating checks to the default values.

Options and Arguments

`-high	-low`	Resets the non-controlling value (high or low) of the clock.
`-hold`	Resets clock gating hold values.
*`inst	hinst	pin	hpin`*	Specifies a list of clocks, instances, or pins for which clock gating checks are to be removed. If this list is not specified, the clock gating check is reset on the current design.
`-rise	-fall`	Resets the specified setup (or hold) value to the rising (or falling) signals on the specified pins.
`-setup`	Resets clock gating setup values.

Examples

- The following command removes the setup value for the clock CLK1:

  ```
  reset_clock_gating_check -setup [{get_clocks CLK1}]
  ```

- The following command removes the hold value for the pin A:

  ```
  reset_clock_gating_check -hold [{get_pins g1/A}]
  ```

Related Information

Related command: set_clock_gating_check on page 1533

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_groups

```
reset_clock_groups
     {-physically_exclusive | -logically_exclusive | -asynchronous}
     {-name name_list | -all}
```

Resets clock groups set using the set_clock_groups command.

Options and Arguments

`-all`	Removes all clock groups in the design.
`-asynchronous`	Specifies to only remove asynchronous clock groups
`-logically_exclusive`	Specifies to only remove logically exclusive clock groups
`-name` *`name_list`*	Specifies to remove previously defined clock groups.
`-physically_exclusive`	Specifies to only remove physically exclusive clock groups

Example

- The following command resets all physically exclusive clock groups in the design:

  ```
  reset_clock_groups -physically_exclusive -all
  ```

- The following command resets the physically exclusive clock group named `CLKGRP0`:

  ```
  reset_clock_groups -physically_exclusive -name CLKGRP0
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_latency

```
reset_clock_latency
     [-source]
     [-clock clock_list]
     {clock | port | pin | hpin}...
```

Resets assertions made by previous `set_clock_latency` commands.

Options and Arguments

`-clock` *`clock_list`*	Specifies the list of clocks from which the associated `set_clock_latency` constraints (at the pin level), are to be removed.
`clock|pin|hpin|port`	Specifies a list of pins, ports, or clock objects for which the assertions are to be removed. **Note:** No clocks can be specified in the object list if the `-clock` option is specified.
`-source`	Removes the source latencies from the specified clocks, pins or ports.

Examples

- The following command removes the clock latency assertion for the clock object in a design:

  ```
  reset_clock_latency [get_clocks WAVE]
  ```

Related Information

Related command: set_clock_latency on page 1539

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_sense

```
reset_clock_sense
     [ -all
     | [-clocks clock_list] pin...]
```

Removes the clock sense constraints specified with the `set_clock_sense` command from the specified pins.

Options and Arguments

`-all`	Removes all specified clock sense constraints.
`-clocks` *`clock_list`*	Specifies the clocks for which the clock sense constraints must be removed. If no clock is specified, the command looks at all clocks defined on each of the pins and ports and changes the clock sense as specified.
`pin`	Specifies the name of the pins for which the clock sense constraints must be removed.

Related Information

Related command: set_clock_sense on page 1543

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_transition

```
reset_clock_transition
     clock_list
```

Removes the clock transition (slew) assertions set on the specified list of clock waveforms.

Options and Arguments

clock_list	Specifies a list of clock waveforms for which the clock transition assertions are to be removed. The `clock_list` variable can be a Tcl list or collection of clock waveforms.

Examples

- The following command removes the clock transition assertion for the clock object in a design:

  ```
  reset_clock_transition [get_clocks WAVE]
  ```

Related Information

Related command: set_clock_transition on page 1547

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_clock_uncertainty

```
reset_clock_uncertainty
     [-from_edge string] [-to_edge string]
     [-from clock_list | -fall_from clock_list | -rise_from clock_list]
     [-to clock_list | -fall_to clock_list |-rise_to clock_list]
     [-hold | -setup]
     [clock|port|instance|pin]...
```

Removes the assertions that were made by previous `set_clock_uncertainty` commands.

Options and Arguments

`{clock | pin | port | instance}`

Removes the target uncertainty assertions from the specified list of clocks or pins. You can specify a list of clock names, or clock root pin or port names relative to the current module. You also can specify a collection of clocks, or clock root pins or ports

Note: You cannot use this parameter when any of the `-from` and `-to` parameter pairs are specified.

`-from clock_list | -fall_from clock_list | -rise_from clock_list`

Removes the assertions for uncertainty value for the specified list of launching clocks.

Note: You must specify one of these parameters with one of the `-to`, `-rise_to`, or `-fall_to` parameters. You cannot use these parameters when pins are specified.

`-from_edge {rise | fall | both}`

Specifies the edge type of the launch clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_from` or `-rise_from`.

`-hold | -setup`

Specifies that the clock uncertainty is removed from hold or setup checks.

If neither option is specified, the uncertainty value is removed from both checks.

`-to clock_list | -fall_to clock_list | -rise_to clock_list`

Removes the uncertainty value from the specified list of capture clocks.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: These options only apply to inter-clock uncertainties.

`-to_edge {rise | fall | both}`

Specifies the edge type of the capture clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_to` or `-rise_to`.

Related Information

Related command:	path_adjust on page 475
Related attributes:	clock_hold_uncertainty
	clock_setup_uncertainty
	hold_uncertainty_by_clock
	setup_uncertainty_by_clock

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_data_check

```
reset_data_check
     [-from {pin|port} | -rise_from {pin|port}
     | -fall_from {pin|port}]
     [-to {pin|port} | -rise_to {pin|port}
     | -fall_to {pin|port}]
     [-setup | -hold]
     [-clock clock_object]
```

Resets data-to-data checks from the specified pins or ports which were set using the set_data_check command. Data-to data checks are non-sequential checks that are designed to constrain a data signal with respect to another data signal.

Options and Arguments

`-clock clock`

Specifies the name of the clock that triggers the data event on the related (`-from`) pin.

In the case where there are multiple data events triggered by different clock phases arriving at the `-from` pin, the `-clock` parameter allows you to specify which data event should be used for the “related pin” side of the timing check.

If this option is not specified, the data check is clock independent.

`-from {pin|port} | -fall_from {pin|port} | -rise_from {pin|port}`

Specifies the pins or ports that are set as related pins for data-to-data checks.

The pin or port list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command.

By default, the `-from` parameter checks both the rising and the falling edges.

Using `-rise_from` parameter checks at the rising edge of the signal on the from pins.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Using -`fall_from` parameter checks at the falling edge of the signal on the from pins.

Note: You cannot use the -`from` parameter to specify multiple objects.

`-setup | -hold`

Specifies whether the data check value is for setup or hold check.

If you do not specify either -`setup` or -`hold` option, the value applies to both setup and hold.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-to {pin|port} |-fall_to {pin|port} |-rise_to {pin|port}`

Specifies the pins or ports that are set as constrained pins for data-to-data checks.

The pin or port list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

If you specify the `-to` option, the data check is performed on the falling and rising edge of specified pin.

Use only one -`to`, -`rise_to`, or -`fall_to` parameter per command.

By default, the -`to` parameter checks both the rising and the falling edges.

Using -`rise_to` parameter checks at the rising edge of the signal on the from pins.

Using -`fall_to` parameter checks at the falling edge of the signal on the from pins.

Related Information

Related command: set_time_unit on page 1641

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_disable_clock_gating_check

```
reset_disable_clock_gating_check
     {pin | hpin | inst | hinst}...
```

Restores the clock gating checks for the specified instances or pins.

Options and Arguments

inst | *hinst* | *pin* | *hpin*

Specifies the list of disabled pins or instances that are to be enabled.

Examples

- The following command restores the check at pin u1/A:

```
reset_disable_clock_gating_check u1/A
```

Related Information

Related command: set_disable_clock_gating_check on page 1556

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_disable_timing

```
reset_disable_timing
     [-from pin] [-to pin] {lib_cell | inst| hinst | lib_pin | pin | hpin |
     port| hport}...
```

Restores the timing arcs that were disabled using the `set_disable_timing` command.

Options and Arguments

`-from pin`
`-to pin`

Specifies the arc(s) that are to be restored. Specify the `-from` and `-to` options together.

You can specify the `-from` and `-to` parameters when the `object_list` contains instance or library cell objects. Do not specify the `-from` and `-to` parameters when the `object_list` includes instance pin or library cell pin objects.s

`{lib_cell | inst | hinst | lib_pin | pin | hpin | port | hport}`

Specifies the instance, instance pin, library arc, library cell, or library cell pin objects.

If the object list includes instance pin or library cell pin objects, all cell arcs attached to the specified pin objects are restored.

If the object list contains instance or library cell objects, and the `-from` and `-to` parameters are not specified, all arcs found in the specified instance or library cell objects are restored.

Examples

- Following example command disables the timing arcs from CK to Q for instance U8, then resets the assertions using the instance reference:

  ```
  reset_disable_timing u8
  ```

Related Information

Related commands: set_disable_timing on page 1558
report timing on page 709

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_drive

```
reset_drive
     {port|port_bus}...
```

Resets the drive resistance on the specified input and/or bidirectional ports of the top cell.

The drive resistance models the cell driving the port. This resistance is used to compute the load dependent delay and the slew at the specified ports.

Options and Arguments

{port | port_bus}	Specifies a list of input or bidirectional ports from which the drive resistance is reset.

Related Information

Related attributes:	external_resistance
	external_driver

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_driving_cell

```
reset_driving_cell
     [-rise] [-fall] [-max | -min]
     port_list
```

Resets the driving cell assertions applied with the `set_driving_cell` command for the given rise or fall, minimum or maximum combination on ports.

If `-rise`, `-fall`, `-min`, or `-max` parameters are not specified, the `reset_driving_cell` command resets the assertion for all rise or fall, minimum or maximum combinations for which the assertion exists.

Options and Arguments

`-fall`	Specifies the falling drive assertions.
`-max`	Specifies the late drive assertion.
`-min`	Specifies the early drive assertion.
`port_list`	Specifies the port(s) to which the driving_cell assertions are reset.
`-rise`	Specifies the rising drive assertions.

Examples

- Following command reset the late drive assertion on the mentioned port:

```
reset_driving_cell -max [vfind -port port_pad_data_in[5]]
```

Related Information

Related command: set_driving_cell on page 1566

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_generated_clock

```
reset_generated_clock
     {-all | clock_list}
```

Removes all the generated clock assertions previously created using the `create_generated_clock` command in the design.

Options and Arguments

`-all`	Removes all generated clock objects.
`clock_list`	Specifies the list of clocks where the generated clock assertion are to be removed.

Examples

- The following command removes the generated clock assertions on all generated clocks

  ```
  reset_generated_clock -all
  ```

- The following command removes all generated clock assertions named GCLK:

  ```
  reset_generated_clock [get_clocks GCLK]
  ```

Related Information

Related command: create_generated_clock on page 1412

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_ideal_network

`reset_ideal_network {pin | port | hnet}...`

Resets the ideal network constraints, specified using the `set_ideal_network` command, on pins, ports, or nets.

Note: The `reset_ideal_network` command is not view or mode-based it applies to all the active views.

Options and Arguments

`{pin | port| hnet}`

Specifies a list of objects - instance pins, ports, or nets.

Related Information

Related command: set_ideal_network on page 1581

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_input_delay

```
reset_input_delay
     [-clock clock]
     [-clock_fall]
     [-level_sensitive]
     [-rise] [-fall] [-max] [-min]
     {port}...
```

Resets previously specified input delay assertions.

Options and Arguments

`-clock clock`	Specifies the reference clock. *Default*: The asynchronous (@) clock If you specify multiple clocks, an error occurs and the constraint is ignored. When you specify the `-clock` parameter, the constraints specified for a pin with matching `-clock` parameter are reset. If you do not specify the `-clock` parameter, all the constraints applied on the specified pin, with or without the `-clock` parameter specification, are reset.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. *Default*: Removes the delay assertions relative to the rising edge.
`-fall`	Resets the input delay for the falling edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.
`-level_sensitive`	Resets output delay that is launched by a positive level sensitive latch. When you specify the `-level_sensitive` parameter, the command resets all the `set_input_delay` constraints specified with the `-level_sensitive` parameter. If `-level_sensitive` parameter is not specified, the constraints specified for the same port, with or without the `-level_sensitive` parameter, are reset. Note: You must specify the `-clock` parameter with this parameter.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max`	Removes the delay assertions that refer to the setup analysis. If you do not specify the `-max` parameter, the input delay assertions for both setup and hold analysis are removed.
`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the input delay assertions for both setup and hold analysis are removed.
`{port}`	Specifies a single or multiple ports for which the input delay is reset. To specify multiple ports, enclose the list with curly braces (`{ }`) and separate the pin information with white space.
`-rise`	Resets the input delay for the rising edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.

Examples

- The following command removes the input delay value on the port:

```
reset_input_delay -clock CLK1 -min [get_ports ABC]
reset_input_delay -clock CLK1 -max [get_ports ABC]
```

Related Information

Related commands: set_input_delay on page 1582

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_load

```
reset_load
     {port | net | port_bus}...
```

Resets the capacitance from the specified ports and nets in the design.

Options and Arguments

{port | net | port_bus}

Specifies the port, net, or port_bus objects from which the capacitance value is reset.

Related Information

Sets these attributes:	external_pin_cap
	external_wire_cap
Related commands:	all_outputs on page 1403
	report port on page 668
	set_drive on page 1563
	set_resistance on page 1636

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_max_capacitance

```
reset_max_capacitance
     {design | port | pin | clock | libpin}...
```

Specifies to reset the maximum capacitance that can be driven by all ports in the specified design(s) or that can be driven by the specified ports.

Resets the `max_capacitance` attribute on the specified design(s) or port(s).

Options and Arguments

clock	Specifies the name of the clock from which the maximum capacitance design rule constraint needs to be removed.
design	Specifies the name of the design from which the maximum capacitance design rule constraint needs to be removed.
libpin	Specifies the name of the library cell pin from which the maximum capacitance design rule constraint needs to be removed.
pin	Specifies the names of the pins from which the maximum capacitance design rule constraint needs to be removed.
port	Specifies the names of the ports from which the maximum capacitance design rule constraint needs to be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_max_fanout

```
reset_max_fanout
     {design | port | libpin}...
```

Resets the maximum fanout load limit constraints from the specified instance pin, ports of the top cell, the specified modules, or lib pins. When the net name is not specified in the constraint, then the constraints is removed from all the nets in the design.

Resets the `max_fanout` attribute on the specified design(s) or port(s).

Options and Arguments

design	Specifies the name of the design from which the maximum fanout design rule constraint is to be removed.
libpin	Specifies the name of the library cell pin from which the maximum fanout design rule constraint is to be removed.
port	Specifies the names of the input ports from which the maximum fanout design rule constraint is to be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_max_time_borrow

```
reset_max_time_borrow
     [clock | instance | pin | port ]...
```

Resets the maximum time that can be borrowed by one stage from the next logic stage which was set using the `set_max_time_borrow` command.

Resets the `latch_max_borrow` attribute.

Options and Arguments

{*clock* | *instance* | *pin* | *port*}

Specifies the object(s) from which the borrow value is reset.

Related Information

Sets this attribute: latch_max_borrow

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_max_transition

```
reset_max_transition
     [design | clock | port | pin | libpin]...
```

Resets the maximum transition design rule for the specified design or port list.

Options and Arguments

clock	Specifies the name of the clock from which the maximum transition design rule constraint needs to be removed.
design	Specifies the name of the design from which the maximum transition design rule constraint needs to be removed.
libpin	Specifies the name of the library cell pin from which the maximum transition design rule constraint needs to be removed.
pin	Specifies the name of the pin from which the maximum transition design rule constraint needs to be removed.
port	Specifies the names of the ports from which the maximum transition design rule constraint needs to be removed.

Related Information

Sets these attributes:	(design) max_transition
	(port) max_transition

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_min_capacitance

```
reset_min_capacitance
     {design | port | clock | pin | libpin}...
```

Resets the minimum capacitance that can be driven by all ports in the specified design(s) or that can be driven by the specified ports.

Options and Arguments

clock	Specifies the name of the clock from which this design rule constraint needs to be removed.
design	Specifies the name of the design from which this design rule constraint needs to be removed.
libpin	Specifies the name of the library cell pin from which the minimum capacitance design rule constraint needs to be removed.
pin	Specifies the name of the pin from which the minimum capacitance design rule constraint needs to be removed.
port	Specifies the names of the ports from which the minimum capacitance design rule constraint needs to be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_min_fanout

```
reset_min_fanout
     {design | port | libpin}...
```

Resets the minimum fanout that set on all input ports in the specified design(s) or the specified input ports.

Note: Genus ignores this command for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the command out for consumption by downstream tools.

Options and Arguments

design	Specifies the name of the design from which the minimum fanout design rule constraint needs to be removed.
libpin	Specifies the name of the library cell pin from which the minimum fanout design rule constraint needs to be removed.
port	Specifies the names of the input ports from which the minimum fanout design rule constraint needs to be removed.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_min_pulse_width

```
reset_min_pulse_width
     [-low] [-high] {pin | port | clock} ...
```

Resets the minimum pulse width checks from the specified objects.

Options and Arguments

`-high`
: Resets the maximum pulse width check from high signal levels only.

`-low`
: Resets the minimum pulse width check from low signal levels only.

Note: If you specify neither the `-low` or the `-high` option, the constraint resets from both the low and high pulses of the clock signals

{pin | port | clock}
: Specifies the list of pins, ports or clocks from which the minimum pulse width constraint must be reset.

Related Information

Related commands:
: report min_pulse_width on page 647
: report timing on page 709

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_min_transition

```
reset_min_transition
     [design | clock | port | pin | lib_pin]...
```

Specifies to reset the minimum transition design rule for the specified design or port list.

Options and Arguments

clock	Specifies the name of the clock from which the minimum transition design rule constraint needs to be reset.
design	Specifies the name of the design from which the minimum transition design rule constraint needs to be reset.
lib_pin	Specifies the name of the library cell pin from which the minimum transition design rule constraint needs to be reset.
pin	Specifies the names of the pins from which the minimum transition design rule constraint needs to be reset.
port	Specifies the names of the ports from which the minimum transition design rule constraint needs to be reset.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_mode

```
reset_mode
     [-type cell ]
     list_of_modes object_list
```

Resets the Liberty timing library modes for a specified instance.

Library modes are set using the set_mode command. By default, all library modes in the timing library are active. If you specify one or more library modes using the `set_mode` command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive. When you reset the library mode for the instance, all library modes become active again.

Options and Arguments

list_of_modes	Specifies the name of the library modes to reset.
object_list	Specifies the instances for which to reset the library modes.
`-type`	Specifies the Liberty cell type of the instance. **Note:** Setting the library mode at the instance level is the only form of `set_mode` that is currently supported. This parameter exists to support constraints coming from external sources.

Related Information

Related command: set_mode on page 1619

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_output_delay

```
reset_output_delay
     [-clock clock]
     [-clock_fall]
     [-level_sensitive]
     [-rise] [-fall] [-max] [-min]
     {port}...
```

Resets previously specified output delay assertions.

Options and Arguments

`-clock clock`	Specifies the reference clock. *Default*: The asynchronous (@) clock. **Note:** If you specify multiple clocks, an error occurs and the constraint is ignored. When you specify the `-clock` parameter, the constraints specified for a pin with matching `-clock` parameter are reset. If you do not specify the `-clock` parameter, all the constraints applied on the specified port, with or without the `-clock` parameter specification, are reset.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. *Default*: Removes the delay assertions relative to the rising edge.
`-fall`	Resets the output delay for the falling edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.
`-level_sensitive`	Resets output delay that is launched by a positive level sensitive latch. When you specify the `-level_sensitive` parameter, the command resets all the `set_output_delay` constraints specified with the `-level_sensitive` parameter. If `-level_sensitive` parameter is not specified, the constraints specified for the same port, with or without the `-level_sensitive` parameter, are reset. Note: You must specify the `-clock` parameter with this parameter.
`-max`	Removes the delay assertions that refer to the setup analysis.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	If you do not specify the `-max` parameter, the output delay assertions for both setup and hold analysis are removed.
`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the output delay assertions for both setup and hold analysis are removed.
`{port}`	Specifies a single or multiple ports for which the output delay is reset. To specify multiple ports, enclose the list with curly braces (`{ }`) and separate the pin information with white space.
`-rise`	Resets the output delay for the rising edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.

Examples

- The following command removes the output delay value on the port:

```
reset_output_delay -clock CLK1 -min [get_ports ABC]
reset_output_delay -clock CLK1 -max [get_ports ABC]
```

Related Information

Related commands: set_output_delay on page 1628

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_path

```
reset_path
     { -all
     | [-from {clock|port|inst|hinst|pin|hpin|port_bus}...]
       [-through {port|inst|hinst|pin|hpin|hnet|port_bus|hpin_bus}...]
       [-to {clock|port|inst|hinst|pin|hpin|port_bus}...]}
     [-setup] [-hold] [-type string]
```

Removes the exceptions of the specified type from the specified paths in the design.

The command removes the exceptions of the specified type from either all paths of from the paths selected with the `-from`, `-through`, and `-to` options.

`reset_path` works only if the expression given with `-from`, `-to` or `-through` **exactly** matches with the original expression OR if the resulting collection of paths of the `reset_path` command are a subset of the collection of paths of original exception.

Options and Arguments

`-all` Removes the specified exception from all paths in the design.

`-from {clock | port | inst | hinst | pin | hpin | port_bus}`
Selects all paths that start from the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-hold` Removes only those exceptions that apply to hold analysis.

`-setup` Removes only those exceptions that apply to setup analysis.

`-through {port | inst | hinst | pin | hpin | hnet | port_bus hpin_bus}`
Selects all paths that pass through the specified points. Specify a Tcl list.

`-to {clock | port | inst | hinst | pin | hpin | port_bus}`
Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-type` *`string`*	Specifies the type of exception to be removed. You can specify one of the following values: `false_path`, `multicycle` or `path_delay`.

Examples

- The following scenario is supported by Genus:

```
set_max_delay -from [get_clocks {clk}] -to [get_pins {current_state_reg[0]/clk}] 5
reset_path -from [get_clocks {clk}] -to [get_pins {current_state_reg[0]/clk}]
```

- The following scenario is not supported by Genus:

```
set_max_delay -from [get_clocks {clk}] 5
reset_path -from [get_clocks {clk}] -to [get_pins {current_state_reg[0]/clk}]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_path_group

```
reset_path_group
     [-all] [-name group_name]
```

Removes the exceptions from the specified path group or completely removes all the groups in the design.

Options and Arguments

`-all`	Removes all path groups.
`-name` *`group_name`*	Removes only the named path group completely. This option cannot be used with the `-all` option.

Examples

- The following command removes the group named GRPA:

  ```
  reset_path_group -name GRPA
  ```

- The following command removes all the path groups

  ```
  reset_path_group -all
  ```

Related Information

Related commands: group_path on page 1463

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_resistance

```
reset_resistance
     net [net]...
```

Resets the wire resistance from the specified nets in the design.

Options and Arguments

net	Specifies a list of nets from which the wire resistance must be removed.

Related Information

Sets these attributes:	(fnet) resistance
	(net) resistance

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_timing_derate

```
reset_timing_derate
     [-cell_delay] [-cell_check] [-net_delay]
     [-early] [-late] [-rise] [-fall]
     [-dynamic] [-static] [-clock] [-data]
     [-delay_corner delay_corner]
     {libcell | instance | design}...
```

This command resets the originally applied derating factors specified on a design or a list of instances (cells, net, or library cells) to the value 1.0. All the derating factors that are set either globally or on specific instances are reset to a default value of 1.0.

Omitting both the `-cell_check` and `-cell_delay` options results in the derating factor being removed from both delay and constraint arcs.

Genus only supports some of the option combinations, but does store all information and writes it out with the `write_sdc` command.

Options and Arguments

`-cell_check`	Removes the derating factors from the constraint arcs.
`-cell_delay`	Removes the derating factors from the cell delay arcs.
`-clock`	Removes the derating factors from the clock paths.
`-data`	Removes the derating factors from the data paths.
`-delay_corner` *`delay_corner`*	Removes the derating factors from this delay corner.
`design`	Specifies the design from which the derating factors must be removed.
`-dynamic`	Resets the specified derating factor for the delta portion of net delays only. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-early`	Removes the derating factors for early paths. Genus only supports this option in combination with the `-data` and `-cell_delay` options.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall`	Removes the derating factors from the fall delays.
`instance`	Specifies the instance(s) from which the derating factors must be removed. Genus does *not* support this option in combination with the `-clock` and `-cell_check` options.
`-late`	Removes the derating factors for the late mode. Genus does *not* support this option in combination with the `-clock` option.
`libcell`	Specifies the library cells from which the derating factors must be removed.
`-net_delay`	Removes the derating factors from the net delays. **Note:** This option is only available in the physical flows such as Genus Physical or Genus Spatial.
`-rise`	Removes the derating factors from the rise delays.
`-static`	Removes the derating factors from the non-delta portion of the net delays only. You can specify this option with the `-cell_delay` and `-net_delay` options.

Related Information

Related command: set_timing_derate on page 1638

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_wire_load_mode

```
reset_wire_load_mode
```

Resets the `wireload_mode` root attribute value.

Related Information

Sets this attribute: wireload_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_wire_load_model

```
reset_wire_load_model
     {design | subdesign | instance | port}...
```

Forces Genus to use the default wire load model for the given design, subdesign, hierarchical instance or port.

Resets the `force_wireload` attribute if the specified object is a design, subdesign or a hierarchical instance. If the specified object is a hierarchical instance, the attribute on the corresponding subdesign is reset.

Resets the `external_wireload_model` attribute if the specified object is a port.

Options and Arguments

design | *subdesign* | *instance*| *port*

Specifies the design, subdesign, hierarchical instance, or port for which you want to reset the wire load model.

If no object is specified, the wire load model is reset from the top-level design.

Related Information

Sets these attributes:	force_wireload
	external_wireload_model
Related attributes:	wireload_mode
	wireload_selection

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reset_wire_load_selection_group

```
reset_wire_load_selection_group
     [-object_list string]
```

Specifies a wire-load selection group.Resets the `wireload_selection` root attribute for the specified object.

Options and Arguments

`-object_list` *`string`*

Specifies the object list from which to reset the selection table.

Related Information

Related attributes: wireload_selection
force_wireload
wireload_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_active_clocks

```
set_active_clocks
     clock...
```

Specifies the clocks that are made active, while others remain inactive. Any clock created after using the `set_active_clocks` command is active by default.

Options and Arguments

clock	Specifies the list of active clocks

Example

- The following command makes the specified clocks active while others remain inactive - only CLK1 is active:

  ```
  set_active_clocks /design/top/clock_domains/m1/CLK1
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_annotated_check

```
set_annotated_check
     -from from_pins
     -to to_pins
     check_value
     [-clock {fall|rise} ]
     [-cond sdf_condition ]
     {-setup | -hold | -recovery | -removal
     | -nochange_high | -nochange_low}
     [-rise] [-fall] [-min] [-max] [-incremental]
```

Annotates the setup, hold, recovery, or removal timing check value between two or more pins on a cell in the current design.

Options and Arguments

check_value
: Specifies timing check value.

-clock {fall|rise}
: Specifies whether the timing check applies to the clock rising or clock falling transition.

  Default: Sets checks for the clock rise and clock fall transitions.

  Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

-cond *sdf_condtion*
: Specifies the condition only if the library has a condition attached to the specified timing check arc.

  The *sdf_condition* must match the syntax of the condition specified in the library.

  Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

-fall
: Specifies that the timing check applies to a data falling transition.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Default: -rise and -fall
`-from` *`from_pins`*	Specifies a list of leaf cell clock pins as starting points of the timing arcs for which timing checks must be annotated.
`-incremental`	Sets the incremental checks. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-max`	Specifies the maximum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (min_max) operating conditions.
`-min`	Specifies the minimum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (min_max) operating conditions.
`-rise`	Specifies that the timing check applies to a data rising transition. *Default*: -rise and -fall
`-setup|-hold|-recovery|-removal|-nochange_high|-nochange_low`	Specifies the timing check type. Make sure there is a corresponding timing arc between the *`from_pins`* and the *`to_pins`* arguments. **Note:** Genus does not support the `-nochange_high` and `-nochange_low` timing checks.
`-to` *`to_pins`*	Specifies a list of leaf cell clock pins as end points of the timing arcs for whcih timing checks must be annotated.

Examples

- The following command annotates a 2.0 setup time between the CP clock pin and the A data pin of the instancename instance:

```
set_annotated_check -setup 2.0 -from instancename/CP -to instancename/A
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_annotated_delay

```
set_annotated_delay
     {-net [-delta_only] | -cell [-cond sdf_condition]}
     [-from from_pins][-to to_pins]
     delay_value [-fall] [-rise] [-min | -max]
     [-increment]
```

Annotates delay to timing arcs. If you specify the set_annotated_delay command without any parameters, the delay value specified is assumed to include both the base delay and any delta (cross-talk) delay.

Options and Arguments

`-cond` *`sdf_condtion`*

Specifies the sdf condition of the selected timing arcs.

Default: The delay applies to all the timing arcs selected using the `-from` and the `-to` options. The expression is ignored for net arcs.

Note: You cannot specify this option with the -net option.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`delay_value`

Specifies the delay value to be annotated to the selected timing arcs.

`-delta_only`

Populates the incremental delay slot with the delay value, so that you can differentiate between the base net delay and a delay offset that is due to crosstalk.

If the base delay was previously calculated or annotated from the SDF, the -delta_only parameter adds the specified delay to the existing base delay.

Note: This parameter can only be used with the -`net` option; you cannot specify it with the -`cell` option.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall`	Specifies that the delay applies to a falling transition. *Default*: `-rise` and `-fall` - The delay applies to both transitions.
`-from` *`from_pins`*	Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected. **Note:** You cannot specify hierarchical pins. If specified, an error messages is displayed.
`-increment`	Increments the specified delay value on top of the existing base delay value. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-max`	Specifies the annotated delay for a particular operating corner. You can apply this option the maximum operating corner.
`-min`	Specifies the annotated delay for a particular operating corner. You can apply this option for the minimum operating corner.
`-net | -cell`	Specifies whether the delay should be annotated to cell arcs or net arcs.
`-rise`	Specifies that the delay applies to a rising transition. *Default*: `-rise` and `-fall` - The delay applies to both transitions.
`-to` *`to_pins`*	Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their source pins specified using the `-from` option and the sink pins specified using this option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected. **Note:** You cannot specify hierarchical pins. If specified, an error message is displayed. **Note:** If `-from` and `-to` options are specified together with the `-cell` option, then the specified pins must belong to the same leaf instance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: If `-from` and `-to` options are specified together with the `-net` option, then the specified pins must be connected to the same net.

This command is not supported on unmapped gates, that is, when the `-cell` option is specified, the instance must be mapped and when the `-net` option is specified driving pin and load pin must belong to mapped instance.

Related Information

Related command: set_annotated_check on page 1525

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_annotated_transition

```
set_annotated_transition
     [-rise] [-fall] [-min] [-max]
     slew {pins|ports}
```

Specifies the transition time to be annotated on the specified pins or ports. This transition time overrides any transition time calculated by Genus.

Note: By default, Genus considers the specified slew as the maximum transition value.

Options and Arguments

`-fall`	Specifies that the transition value is for a falling transition time. **Note:** By default, the transition value is set for both falling and rising transition times.
`-max`	Specifies that the transition value is specified for a maximum transition time. **Note:** The maximum values are used for timing analysis.
`-min`	Specifies that the transition value is specified for a minimum transition time. **Note:** The minimum values are not used for timing analysis, but they will be written out when you execute the `write_sdc` command.
`{pins|ports}`	Specifies the pins or ports to be annotated.
`-rise`	Specifies that the transition value is for a rising transition time. **Note:** By default, the transition value is set for both falling and rising transition times.
slew	Specifies the transition value to be annotated to the specified pins and ports.

Example

The following command sets the falling and rising (maximum) transition time of port A to 0.6.

```
legacy_genus:/designs/rct> set_annotated_transition 0.6 [get_ports A]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_case_analysis

```
set_case_analysis
     string {port | pin}...
```

Sets constant or transitional values to a list of pins or ports for use by the timing engine. Sets the `timing_case_logic_value` attribute of the specified pin or port to the specified logic value when you do not perform multi-mode timing analysis.

When performing multi-mode timing analysis, this command *updates* the `timing_case_logic_value_by_mode` attribute of the specified pin or port with the specified logic value for the current mode. If the attribute value already contained a value for the specified mode, it will overwrite the existing value. If the attribute value did not yet contain a value for the specified mode, it will add the mode name and the corresponding logic value.

When you use this constraint in an SDC file, the constraint will be applied to the mode specified with the `-mode` option of the read_sdc command that reads in the file.

Options and Arguments

{port|pin}	Specifies the list of pins or ports to which the logic value or transition value applies.
string	Applies any of the following values to the specified pin or port. {0 | 1 | zero | one} — A logic value of 0 or 1 is propagated through combinational logic to disable or enable parts of logic {rising | falling | rise | fall} — Specifies a valid transition value to be considered by the timing engine. For example, if a rising value is specified at a certain pin or port, only the rising signal transition from the pin or port is considered for timing analysis and the opposite transition type (falling) is ignored.

Related Information

Related attributes:	(pin) timing_case_logic_value
	(pin) timing_case_logic_value_by_mode
	(port) timing_case_logic_value
	(port) timing_case_logic_value_by_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_exclusive

```
set_clock_exclusive
     {-group string}... -pin {pin|hpin}
```

Controls filtering of mutually-exclusive clocks at strategic points in the design, for example, the output of clock multiplexers.

Options and Arguments

`-group string`	Specifies the clock group. Usage: `-group clock_list_1 -group clock_list_2` At least one `-group` is required .
`-pin {pin|hpin}`	Specifies the pin or hpin name.

Example

- Usage example:

```
set_clock_exclusive \
-group {/design/top/clock_domains/m2/CLK1 /design/top/clock_domains/m1/CLK2}\
-group {/design/top/clock_domains/m3/CLK3 /design/top/clock_domains/m4/CLK4}\
-pin [vfind . -pin *U1/Y]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_gating_check

```
set_clock_gating_check
     [-setup float] [-hold float]
     [-rise | -fall]
     [-high] [-low]
     [design | clock | instance | pin]...
```

Performs a setup check with respect to the edge of the clock signal that changes the state of the clock pin from controlling to non-controlling.

For AND and NAND gates, the setup check is performed with respect to the rising edge of the clock input. For OR and NOR gates, the setup check is performed with respect to the falling edge of the clock input.

If several clock gating checks are specified, the clock gating check specified on a pin has a higher priority then the clock gating check specified on an instance, followed by the clock gating check specified on a clock, a subdesign, the design.

Note: This command is not supported on the clock pins of an instance.

Note: The clock gating constraint is not propagated along the clock tree.

The setup check is performed with respect to the edge of the clock signal that changes the state of the clock pin from controlling to non-controlling. For example, for AND and NAND gates, the setup check is performed with respect to the rising edge of the clock input. For OR and NOR gates, the setup check is performed with respect to the falling edge of the clock input.

The hold check is performed with respect to the edge of the clock signal that changes the state of the clock pin from non-controlling to controlling. For example, for AND and NAND gates, the hold check is performed with respect to the falling edge of the clock input. For OR and NOR gates, the hold check is performed with respect to the rising edge of the clock input.

If the gate logic in unknown, setup and hold checks are performed with respect to both the rising edge and the falling edge of the clock signal and the timing analyzer uses the worst case values.

Timing analysis uses the following priority rules to determine the setup (or hold) value to use for a particular clock gating check:

- Use the clock gating check assertion on the data pin if it exists.
- Otherwise, use the clock gating check assertion on the clock pin if it exists.
- Else, use the clock gating check assertion on the clock waveform for the clock pin if it exists.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Otherwise, use the clock gating check assertion, if it exists.
- If no clock gating check assertion is present, use the default setup (or hold) value of 0.0.

Note: This command has no effect on a clock gating check defined as a non-sequential setup or hold check in the library and reported as `ClockGatingSetup` or `ClockGatingHold`.

Options and Arguments

`[-fall | -rise]`

Applies the specified setup (or hold) value to the rising or falling signals on the specified pins. If the clock is specified or the object_list variable is not specified, the value applies to rising or falling, setup or hold.

If neither option is specified, the clock gating check is applied to both the rising and falling transitions.

`[-high | -low]`

Specifies the non-controlling value of the clock is high or low.

Timing analysis performs the check on the non-controlling value of the clock.

By default, the software determines the non-controlling value of the clock using information from the logic of the gate.

For OR gates the non-controlling value is low. For complex gates, such as XOR and MUX, the clock is never controlling and it does not perform checks unless you specify either the `-high` or `-low` option. The specified value takes precedence over the default values. This option can only be used with pins or instances.

`-hold` *`float`*

Specifies the hold value.

Default: 0.0

Note: Because the tool does not perform any minimum analysis, Genus just stores this value and writes it out when the `write_sdc` command is given.

`design | clock | instance | pin`

Specifies the design, module, clock, instance, or pin for which you want to set the clock gating check.

If a clock is specified, the check applies to all the clock gating gates driven by this clock.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

If an instance is specified, the check applies to all input pins of the instance.

If no object is specified, the clock gating check is applied to the current design.

`-setup` *`float`*

Specifies the setup value.

Default: `0.0`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_groups

```
set_clock_groups
     -name string
     [ -asynchronous [-allow_paths]
     | -logically_exclusive | -physically_exclusive ]
     -group clock_list [-group clock_list]... [-comment string]
```

Specifies the groups of clocks between which timing analysis will not be done (does a `set_false_path` between the groups of clocks).

Note: This command was added in SDC 1.7.

Options and Arguments

`-allow_paths`	Allows paths between asynchronous clock groups to be traced in timing analysis. These paths are blocked by default. When you use this option, the software still considers the paths to be asynchronous for signal integrity considerations. You can only use this option with the `-asynchronous` option. This option is not supported for signal integrity analysis. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-asynchronous`	Defines clock group with asynchronous clocks. Asynchronous clocks have no specified phase relationship. The tool considers asynchronous nets as false paths for timing analysis. Use the `-allow_paths` option to trace these paths in timing analysis. This option is not supported for signal integrity analysis.
`-comment` *`string`*	Specifies a comment to be tagged to this command.
`-group` *`clock_list`*	Specifies a group of clocks. Specify at least one group. **Note:** A clock can belong to one group only. **Note:** If the number of clocks in a group equals the number of clocks in the design, no false paths are applied.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-logically_exclusive`

Defines clock group with clocks that do not have any functional paths between them, but coexist in the design at the same time. The tool considers aggressor interactions between these clocks as false paths for timing analysis.

This option is not supported for signal integrity analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-name` *`string`* Specifies the name of the clock group.

`-physically_exclusive`

Defines clock group with clocks that cannot exist in the design at the same time. The tool does considers these clock groups as false paths for timing analysis.

This option is not supported for signal integrity analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Examples

- The following command sets a false path between clock1 and clock2, and clock1 and clock3.

  ```
  set_clock_groups -name cg_group1 -logically_exclusive -group {clock1} \
  -group {clock2 clock3}
  ```

  When executing the `write_sdc` command, the `set_clock_groups` command in the input SDC file is written out verbatim. The false path exception created by the `set_clock_groups` command is not written out.

- The following command sets a false path between clock1 and clock2, and clock1 and clock3.

  ```
  set_clock_groups -name cg_group1 -logically_exclusive -group {clock1} \
  -group {clock2 {clock3}}
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command results in an error because clock `clk1` is defined in two groups.

```
legacy_genus:/> set_clock_groups -group {clk1 clk2} -group {clk1 clk3}
Error   : Invalid SDC command option combination. [SDC-204] [set_clock_groups]
        : /designs/top/timing/clock_domains/domain_1/clk1 can be specifed as
part of only one group in 'set_clock_groups'
        : This option is not valid for the indicated SDC command. Check the SDC
command and contact Cadence customer support if you believe this option
combination should be supported.
```

- The following command will not apply any false paths constraints assuming that the designs has only those three clocks.

```
set_clock_groups -group { clk1 clk2 clk3} -comment "comment for clock_group"
```

Related Information

Related attribute: (exception) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_latency

```
set_clock_latency
     [-min ] [-max]
     [-rise ] [-fall]
     [-early] [-late]
     [-source] [-clock clock_list] [-clock_gate]
     latency {clock | port | pin}...
```

Specifies a clock source or network, early or late latency value of one or more clocks that propagate to the specified pin or port.

When you specify this command, it sets a pin, port or clock latency-related attribute.

- Clock network latency is the time taken by clock signal from definition point to sequential logic clock pin.
- Clock source latency is the source latency defined as time taken by a clock signal from clock source (PLL) to clock definition point

Clock value has two components - source value and network value.

Source value:

- Specified using the `-source` parameter and is the delay from the external clock generator to the clock port of the design.
- Applied when the clock is either ideal or propagated.
- `set_clock_latency` set on a clock port has higher precedence than `set_clock_latency` set on a clock waveform.
- Specified on clock waveforms with both the `set_input_delay` command and the `set_output_delay` command.

Network value (clock tree delay):

- Default value type and is the delay from the clock port to the register.
- Applied only when the clock is ideal. When the clock is propagated, this delay is replaced by the actual clock tree delays.
- Specified on a clock waveform or a clock port is ignored if the clock is propagated.
- `set_clock_latency` set on a timing pin has higher precedence than `set_clock_latency` set on a waveform.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- If there are multiple `set_clock_latency` and `set_propagated_clock` assertions between the clock waveform, clock root and clock endpoint, the closest assertion to the clock endpoint takes precedence.
- Specified on internal circuit pins or hierarchical ports and used in the fanout cone of the network value pin, when a clock is ideal.

Use the `set_clock_latency` command in two ways:

- To specify clock value for a clock waveform:
  - Use the following syntax for source value associated with a clock signal:
    `set_clock_latency [-source] [-early|-late] [-min][-max] \ value obj_list`
  - Use the following syntax for network value associated with a clock signal:
    `set_clock_latency [-min] [-max] value obj_list`
- You cannot specify the `-late` and `-early` parameters for network value. Specifying value on a port or pin overrides any previous value specified on the port pin or waveform.
- The *`obj_list`* argument is the list of the clock waveform names as specified by the `create_clock` command.

Options and Arguments

{clock | pin | hpin | port}

Specifies a list of clock waveform names or a list of pins to associate with the clock value. The pin can be a port or an instance pin.

-clock clock_list

Associates clock value constraints, placed at the pin level, with a specific clock. This makes it possible to specify a different value per clock.

If no clocks are specified, the latency value applies to all clocks that propagate to the specified pin or port.

Note: You cannot use the `-clock` parameter with clock waveform objects.

-clock_gate

Allows setting the local clock value on pin or port objects.

This setting overwrites the existing value of clock gating cells and is used when performing timing checks against the specified pins or ports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-early | late`	Specifies clock arrival time with respect to the early or the late time of the clock signal. In setup analysis, launch path is the late path and capture path is the early path In hold analysis, launch path is the early path and the capture path is the late path. If neither parameter is specified, the default is both early and late. Use these parameters only with the `-source` parameter. **Note:** You cannot use these parameters with network value. Specify the clock value at the given operating corner. **Note:** The `-min` and `-max` parameters refer to the operating corner and the `-early` and `-late` parameters refer to the clock arrival time at the source. Use the `-early` and `-late` parameters with the `-source` parameter to specify the four different clock latencies
`-fall`	Indicates that the specified latency value applies to a fall edge.
`latency`	Specifies the latency value. This is a floating number.
`-max`	Specifies the clock value for a particular operating corner. Apply the `-max` parameter to the maximum operating corner.
`-min`	Specifies the clock value for a particular operating corner. Apply the `-min` parameter to the minimum operating corner.
`-rise | fall`	Specifies the `-rise` or `-fall` parameter to the waveform at register clock pins for ideal value and to the waveform at the source for source value. Ideal network value does not change polarity when crossing negative unate arcs.
`-source`	Specifies a source value. If this option is not specified, the specified latency value is considered to be a network latency.

Examples

- The following commands specify the launch latency for hold analysis of the rise edges of all clocks propagating to port `p1`.

```
set_clock_latency -early -min -rise 0.55 [find / -port p1]
set_attribute clock_network_early_latency \
{no_value no_value 550 no_value} [find / -port p1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following commands specify the capture latency for hold analysis of the rise edge of all clocks propagating to port `p1`.

```
set_clock_latency -late -min -rise 0.59 [find / -port p1]
set_attribute clock_network_early_latency \
{590 no_value no_value no_value} [find / -port p1]
```

- The following commands specify the capture latency for setup analysis and launch latency for hold analysis for the rise edges of the clocks propagating to port `p1`.

```
set_clock_latency -early -rise 0.65 [find / -port p1]

set_attr clock_network_early_latency {no_value no_value 650 no_value} \
[find / -port p1]
set_attr clock_network_late_latency {650 no_value no_value no_value} \
[find / -port p1]
```

- The following commands specify the capture latency for hold analysis and launch latency for setup analysis for the rise edges of the clocks propagating to port `p1`.

```
set_clock_latency -late -rise 0.74 [find / -port p1]

set_attr clock_network_early_latency {740 no_value no_value no_value} \
[find / -port p1]
set_attr clock_network_late_latency {no_value no_value 740 no_value} \
[find / -port p1]
```

Related Information

Related command:	remove_clock_latency on page 1472
Related attributes:	clock_network_early_latency
	clock_network_late_latency
	clock_source_early_latency
	clock_source_early_latency
	network_early_latency_by_clock
	network_late_latency_by_clock
	source_early_latency_by_clock
	source_late_latency_by_clock
	use_multi_clks_latency_uncertainty_optimize
	use_multi_clks_latency_uncertainty_report

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_sense

```
set_clock_sense
     [-clocks clock_list]
     [ -positive | -negative | -pulse string
     | -logical_stop_propagation | -stop_propagation]
     {pin|hpin|port}...
```

Specifies clock phase or polarity with respect to source clock from specified pin (forward direction only). Selects which phase of the clock to filter at the specified point.

Note: .This command was added in SDC 1.7.

Options and Arguments

`-clocks clock_list`

Specifies the list of clocks passing through the specified point to which the constraint applies.

If no clock is specified, the command looks at all clocks defined on each of the pins and ports and changes the clock sense as specified.

`-logical_stop_propagation`

Stops the propagation of the specified clocks through the specified pins.

`-negative`

Specifies that the specified pins and ports are inverted sources of the clock.

If the pin or port is a non inverting source of the clock, the clock sense on the port is changed and it becomes an inverting source of the clock.

`{pin|hpin|port}`

Specifies the list or collection of cell output pins. The unateness specified by the set_clock_sense command is applied to clocks propagating through the specified point.

`-positive`

Specifies that the specified pins and ports are sources of the clock.

If the pin or port is an inverting source of the clock, the clock sense on the port is changed and it becomes a non-inverting source of the clock.

`-pulse string`

Specifies the type of pulse clock applied.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-stop_propagation` Stops the propagation of the specified clocks and clock source data paths through the specified pins.

Examples

- The following command stops the propagation of clock CLK1 through the pin `CLKBUF1/Z`:

```
set_clock_sense -logical_stop_propagation -clocks CLK1 CLKBUF1/Z
```

- The following command allows the positive phase clock CLK1 to pass through CLKBUF1/Z and filters out the negative phase clock CLK1:

```
set_clock_sense -positive -clocks CLK1 CLKBUF1/Z
```

- The following command stops clock phases starting from pin1:

```
set_clock_sense pin1 -logical_stop_propagation
```

Related Information

Sets these attributes: clock_sense_negative, clock_sense_positive, clock_sense_stop_propagation, propagated_clocks

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_skew

```
set_clock_skew
     [-minus_uncertainty] [-plus_uncertainty]
     float {clock | port}...
```

Sets the skew on the specified clocks and ports.

Skew is internally modeled as clock uncertainty. If neither `-minus_uncertainty` or `-plus_uncertainty` are specified, the skew is applied for both setup and hold checks. However, only the setup value is used for timing analysis and optimization, while the hold value is stored and written out with the `write_sdc` command.

Options and Arguments

`{clock | port}`
: Specifies the clocks and ports to which the specified skew value applies.

`float`
: Specifies the skew value.

`-minus_uncertainty`
: Shifts the clock edge to the left to make the clock arrive earlier.

  Used to check setup delays.

  Equivalent to:

  `set_clock_uncertainty -setup` *`uncertainty`* `[`*`clock`*`|`*`port`*`]...`

`-plus_uncertainty`
: **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

  Equivalent to:

  `set_clock_uncertainty` *`uncertainty`* `[`*`clock`*`|port]...`

Example

The following command sets the skew for clock `clk1` to 0.5 time units.

```
legacy_genus:/> set_clock_skew 0.5 [find / -clock clk1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command:	set_clock_uncertainty on page 1548
Sets these attributes:	clock_hold_uncertainty
	clock_setup_uncertainty

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_transition

```
set_clock_transition
     {-min | -max} {-rise | -fall} float clock_list
```

Specifies the clock transition (slew) on the specified clocks.

You can specify four slew values: the minimum rise, minimum fall, maximum rise, and maximum fall slew.

The minimum values are not used for timing analysis, but they will be written out when you execute the `write_sdc` command.

Options and Arguments

clock_list	Specifies a list of clocks only with the ideal mode in the design, which affects the slew times on all register clock pins in the transitive fanout of the specified clocks. Specified `set_clock_transition` values on register clock pins in the fanout of a clock with propagated latency are ignored.
`-fall`	Specifies the falling edge (transition value) of the register clock pins on which the slew time is asserted.
float	Specifies the slew value for the specified clock(s).
`-max`	Indicates that the specified value is a maximum slew value.
`-min`	Indicates that the specified value is a minimum slew value.
`-rise`	Specifies the rising edge (transition value) of the register clock pins on which the slew time is asserted.

Related Information

Related command:

remove_clock_transition on page 1473

set_clock_latency on page 1539

set_clock_uncertainty on page 1548

set_input_delay on page 1582

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_clock_uncertainty

```
set_clock_uncertainty
     [-from_edge string] [-to_edge string]
     [-from clock_list | -fall_from clock_list |
       -rise_from clock_list]
     [-to clock_list | -fall_to clock_list |
       -rise_to clock_list]
     [-hold | -setup]
     uncertainty
     [clock|port|instance|pin]...
```

Specifies the uncertainty on the clock network. You specify either a simple/intra-clock or an inter-clock uncertainty.

- Simple/intra-clock uncertainties are defined directly on a clock, port, pin, or instance. These uncertainty values are stored in attributes. When you specify a clock, it means that the uncertainty applies to all sequential elements driven by that clock. When applied on a port or pin, it applies to all clocks (and their corresponding sequential elements) in the fanout of the port or the pin.
- The inter-clock uncertainties are defined using options such as -from, -to, -rise_from, -rise_to. These are modeled as path_adjust exceptions. These uncertainties take precedence over the simple uncertainty values. When you specify inter-clock uncertainty, it must be specified to cover all source and destination clock combinations. For example, if you specify paths from CLK1 to CLK2, you must specify the uncertainty for CLK2 to CLK1 even if the values are the same.

Options and Arguments

{clock | pin | port | instance}

Specifies the clocks, pins, ports and instances to which the specified uncertainty value applies. In case of instances, the uncertainty is applied on the input pins of the instance.

Note: These arguments only apply to simple uncertainties.

-from clock_list | -fall_from clock_list | -rise_from clock_list

Applies the uncertainty value to the specified list of launching clocks.

Note: These options only apply to inter-clock uncertainties.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-from_edge {rise | fall | both}`

Specifies the edge type of the launch clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_from` or `-rise_from`.

`-hold | -setup`

Specifies that the clock uncertainty applies to hold or setup checks.

If neither option is specified, the uncertainty value applies to both checks.

Note: These options apply to both types of uncertainties.

`-to` *`clock_list`* `| -fall_to` *`clock_list`* `| -rise_to` *`clock_list`*

Applies the uncertainty value to the specified list of capture clocks.

Note: These options only apply to inter-clock uncertainties.

`-to_edge {rise | fall | both}`

Specifies the edge type of the capture clock.

Note: This option only applies to inter-clock uncertainties and cannot be specified with either `-fall_to` or `-rise_to`.

`uncertainty`

Specifies the uncertainty value for the clock(s). Use a floating number.

Examples

- The following command sets a (simple) setup uncertainty of 0.4 sdc units for all paths ending at `reg1` and captured by the rising transition of all clocks propagating to `reg1/CK`.

```
set_clock_uncertainty -setup -rise 0.4 [find / -pin reg1/CK]
```

- The following command sets a (simple) setup uncertainty of 0.4 sdc units for all paths ending at `reg1` and captured by the rising transition of `clk1`.

```
set_clock_uncertainty -setup -rise -clock clk1 0.4 [find / -pin reg1/CK]
```

- The following command sets a (inter clock) setup uncertainty of 0.5 sdc units for all paths launched by `clk2` and captured by `clk1`.

```
set_clock_uncertainty -setup -from clk2 -to clk1 0.5
```

- The following example specifies that all paths leading to registers or ports clocked by CLKB have setup uncertainty of 0.5 and hold uncertainty of 0.2:.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
set_clock_uncertainty -setup 0.5 [get_clocks CLKB]
set_clock_uncertainty -hold 0.2 [get_clocks CLKB]
```

If there are paths from other clocks to CLKB, then unless otherwise specified, the intra-clock uncertainty also models the inter-clock uncertainty.

- The following example specifies inter-clock uncertainties between CLKA and CLKB clock domains:

```
set_clock_uncertainty 1.5 -from CLKA -to CLKB
set_clock_uncertainty 1.5 -from CLKB -to CLKA
```

- The following example specifies inter-clock uncertainties between the CLKA and CLKB clock domains with specific edges:

```
set_clock_uncertainty 0.25 -rise_from CLKA -to CLKB
set_clock_uncertainty 0.25 -fall_from CLKA -rise_to CLKB
set_clock_uncertainty 1.5 -from CLKA -fall_to CLKB
```

- The following example specifies that when both inter-clock and intra-clock uncertainties are specified, then inter-clock takes a higher precedence:

  Intra-clock and Inter-clock Uncertainty Conflict

```
set_clock_uncertainty 0.6 -from CLKB -to CLKA
set_clock_uncertainty 0.5 [get_clocks CLKA]
```

  In the example above the two uncertainty values on Clock CLKA are different. For all paths from CLKB to CLKA, the inter-clock uncertainty takes precedence, and a value of 0.6 is used. For all other paths where the destination clock is CLKA, the intra-clock value of 0.5 is used.

Related Information

Affects this command:	path_adjust on page 475
Sets these attributes:	clock_hold_uncertainty
	clock_setup_uncertainty
	hold_uncertainty_by_clock
	setup_uncertainty_by_clock

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_constraint_mode

```
set_constraint_mode
     mode
```

Specifies the mode to which subsequent constraints—entered *interactively* at the Genus prompt—must be applied.

Specify this command first followed by the changes in the constraints for that specified mode.

Once this command is set, all constraints that need a mode specification will apply the constraint to the specified mode.

Note: You do not need to explicitly set `set_constraint_mode` before each constraint.

Tip

When reading in the SDC file using the `read sdc` command, use the `-mode` option to specify the mode to which all the constraints in the SDC file must be applied.

Options and Arguments

mode	Specifies the mode to which subsequent constraints must be applied.

Example

- The following command shows the error messages issued when using interactive constraint mode:

```
set_false_path -to delay_reg[1]
Error   : SDC command requires a constraint mode specification. [SDC-235]
          [retrieve_mode]
        : The 'set_false_path' command requires a constraint mode to be
          specified when in MMMC.
        : The indicated command requires that a constraint mode be specified.
          Use the 'set_interactive_constraint_modes' to indicate the
          constraint mode for the command.
```

When `set_constraint_mode` command is used the constraint is applied:

```
set_constraint_mode synth_setup_ss
mode:top/synth_setup_ss
set_false_path -to delay_reg[1]
exception:top/syn_setup_ss/dis_1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_data_check

```
set_data_check
     [-from {pin|port} | -rise_from {pin|port}
     | -fall_from {pin|port}]
     [-to {pin|port} | -rise_to {pin|port}
     | -fall_to {pin|port}]
     [-setup | -hold]
     [-clock clock_object] value
```

Sets data-to-data checks using the specified values of setup and hold time. Data-to data checks are non-sequential checks that are designed to constrain a data signal with respect to another data signal.

In a normal sequential check, a data signal is constrained with respect to a clock signal. In these cases single-cycle timing is used - allowing one full clock cycle for the data transfer. In a non-sequential data-to-data check, there is no clock reference. Therefore zero-cycle timing rules are used. You can use the `set_multicycle_path` constraint to adjust the cycle accounting rules.

For setup checks the tool computes the maximum arrival time on the constrained pin (specified with the `-to|-rise_to|-fall_to` option) and checks it against the minimum arrival time on the related pin (specified with the `-from|-rise_from|-fall_from` option).

The arrival times on the related pin can depend on the clock that triggers the data event on this pin.

Current Limitations

1. Genus does not support data-to-data checks on top-level ports. Genus will not infer a check and this will also not be captured as part of `write_sdc`.
2. Genus does not support data-to-data checks specified on pins belonging to different instances. Genus will not infer such a data check.
3. Genus also does not support specifying `set_data_check -from/-to` instances.

Options and Arguments

`-clock` *`clock`*	Specifies the name of the clock that triggers the data event on the related (`-from`) pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

In the case where there are multiple data events triggered by different clock phases arriving at the `-from` pin, the `-clock` parameter allows you to specify which data event should be used for the “related pin” side of the timing check.

If this option is not specified, the data check is clock independent.

`-from {pin|hpin|port} | -fall_from {pin|hpin|port} | -rise_from {pin|hpin|port}`

Specifies the pins or ports that are set as related pins for data-to-data checks.

The pin or port list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command.

By default, the `-from` parameter checks both the rising and the falling edges.

Using `-rise_from` parameter checks at the rising edge of the signal on the from pins.

Using `-fall_from` parameter checks at the falling edge of the signal on the from pins.

Note: You cannot use the `-from` parameter to specify multiple objects.

`-setup | -hold`

Specifies whether the data check value is for setup or hold check.

If you do not specify either `-setup` or `-hold` option, the value applies to both setup and hold.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-to {pin|hpin|port} |-fall_to {pin|hpin|port} |-rise_to {pin|hpin|port}`

Specifies the pins or ports that are set as constrained pins for data-to-data checks.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

The pin or port list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

If you specify the `-to` option, the data check is performed on the falling and rising edge of specified pin.

Use only one -`to`, -`rise_to`, or -`fall_to` parameter per command.

By default, the -`to` parameter checks both the rising and the falling edges.

Using -`rise_to` parameter checks at the rising edge of the signal on the from pins.

Using -`fall_to` parameter checks at the falling edge of the signal on the from pins.

value Specifies the value of the setup or hold time for the check, in the time units specified with the `set_time_unit` command. Use a floating number.

Example

- Consider the following module:

```
module top (in1,in2,c1,c2,c3,out);
    input in1,in2,c1,c2,c3;
    output out;
    fflopd inst1 (.D(in1),.CK(c1),.Q(w1));
    fflopd inst2 (.D(in2),.CK(c2),.Q(w2b));
    buf1 b1 (.A(w2b),.Y(w2b1));
    buf1 b2 (.A(w2b1),.Y(w2b2));
    buf1 b3 (.A(w2b2),.Y(w2b3));
    buf1 b4 (.A(w2b3),.Y(w2b4));
    buf1 b5 (.A(w2b4),.Y(w2b5));
    buf1 b6 (.A(w2b5),.Y(w2b6));
    buf1 b7 (.A(w2b6),.Y(w2b7));
    nand2 inst3 (.A(w1),.B(w2b7),.Y(w3));
    fflopd inst4 (.D(w3),.CK(c3),.Q(out));
endmodule
```

- The following command specifies a setup check between pins `B` and `A` of instance `inst3` and verifies that the arrival time on `A` is less than the specified value (0.9 time units):

```
set_data_check -setup -from inst3/B -to inst3/A 0.9
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: set_time_unit on page 1641

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_disable_clock_gating_check

```
set_disable_clock_gating_check
     [-setup] [-hold]
     [-rise] [-fall]
     [-high | -low]
     {pin|instance} ...
```

Disables the clock gating check inferred on the specified pins or the enable pins of the specified instance.

Note: This command is supported only on mapped instances and pins of mapped instances. This command is not supported on the clock pins of an instance.

Note: To write this constraint in the output SDC file, you must set the following variable:

```
set write_disable_and_rmdisable_cgc 1
```

Options and Arguments

`-fall`	Disables only clock gating checks specified for a falling clock transition.
`-high`	Disables any clock gating check for which the non-controlling value of the clock is high.
`-hold`	**Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
instance	Disables all clock gating checks inferred at every single enable pin of the specified instance. To restore the clock gating check on an instance, use: `remove_disable_clock_gating_check` *instance*
`-low`	Disables any clock gating check for which the non-controlling value of the clock is low.
pin	Specifies the pin on which to apply the command. If the pin is the enable pin of a clock gate, all clock gating checks inferred at the enable pin are disabled and the timing path is not broken at this pin.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

To restore the clock gating check on a pin, use:

`remove_disable_clock_gating_check` *`pin`*

`-rise`

Disables any clock gating check specified for a rising clock transition.

`-setup`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Related Information

Related command: remove_disable_clock_gating_check on page 1474

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_disable_timing

```
set_disable_timing
     { [-from pin] [-to pin] {libcell|instance}...
     | {libpin|pin|port|subport}... }
```

Disables the timing propagation through the specified objects.

- For a pin of an unmapped instance, Genus inserts a buffer driving this pin and disables the input to output timing arc of this buffer.
- For a pin, port, subport of a hierarchical instance, Genus breaks the timing paths at this point.

  For a subport, the timing path is broken at the pin on the other side of the subport.

Options and Arguments

`-from pin`
: Disables the timing arcs starting from this pin in the specified libcells or instances.

  Note: An error is given when you specify an object type other than a libcell or instance with this option.

`{libpin | pin | port | subport}`
: Specifies the libpin, pin, port, or subport at which the timing path(s) must be broken.

`{libcell | instance}`
: Specifies the libcells or instances for which you want to disable timing arcs. If you specify a libcell or instance without a `-from` or `-to` option, the tool disables all timing arcs for that libcell or instance.

`-to pin`
: Disables the timing arcs ending at this pin in the specified libcells or instances.

  Note: An error is given when you specify an object type other than a libcell or instance with this option.

Examples

- The following command disables the timing arcs between pins A and Z in instances i1 and i2.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
set_disable_timing -from A -to Z {i1 i2}
```

- The following command disables the outgoing timing paths from pin Z of instance i1.

```
set_disable_timing {get_pins i1/Z}
```

Related Information

Related command: remove_disable_timing on page 1476

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_dont_touch

```
set_dont_touch
     {design | subdesign | instance| net | libcell} ...
```

Prevents the specified object from being modified during optimization. However, the specified object can be moved during optimization.

This command does not work on unmapped or unresolved objects.

When you specify a net, it cannot be buffered or become a part of the logic cone restructuring. However, you can resize the instances connected to this net because the `dont_touch` attribute set of the net does not propagate on the connected instances.

For net objects, the `preserve` attribute is only set on the specified net, and not on all the segments of the physical net (if the net traverses many hierarchies).

To preserve the entire *hierarchical* net, set the following variable:

```
set sdc_set_dont_touch_physical_net 1
```

Options and Arguments

`{design | subdesign | instance | net | libcell}`

Specifies the list of instances, nets, library cells, or subdesigns that you want to preserve.

Examples

- The following command prevents the specified instance from being modified:

  ```
  set_dont_touch [get_cells {TWA/FF1}]
  ```

- The following command prevents the specified instance explicitly from being modified:

  ```
  set_dont_touch /dtmf_recvr_core/DMA_INST true
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_dont_touch_network

```
set_dont_touch_network
     [-no_propagate]
     {clock | port | pin | pin_bus | port_bus }...
```

Preserves the nets and combinational logic fanning out from a particular port or pin. If a clock object is specified, the nets and combinational logic fanning out from the clock sources are preserved.

Note: If a net segment has any unmapped leaf-level instances, unresolved instances, or instances which are not logical hierarchies (such as a bmux), the command will preserve the net segment leading up to the first unmapped leaf combinational instance or non-logical hierarchical instance.

Note: `set_dont_touch_network` can have unintended consequences by limiting optimization. For example, using this command on an input port will results in a `dont_touch` on the first sequential element and the logic will not be optimized. This happens if the `-no_propagate` switch is not used.

Options and Arguments

{*clock* | *port* | *pin* | *pin_bus* | *port_bus*}	Specifies the objects to be preserved.
`-no_propagate`	Preserves all the nets driven by the specified pin up to the first leaf-cell-input (combinational or sequential). If this option is omitted, preserving stops at the first sequential cell in the network.

Example

- The following command preserves all nets and combinational logic fanning out of port `in`, till the first sequential cell is encountered in each fanout path.

  ```
  set_dont_touch_network [find / -port in]
  ```

Related Information

Sets this attribute: preserve

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_dont_use

```
set_dont_use
     libcell_list
```

Indicates whether the specified library cell(s) should be avoided by the technology mapper. Sets the `avoid` attribute on the specified cells to the specified value.

Options and Arguments

`libcell_list`	Specifies the library cell(s) that must be marked for the technology mapper.

Examples

- The following two SDC commands result in the `avoid` attribute being set to `true` on the `inv1` cell.

  ```
  set_dont_use inv1
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_drive

```
set_drive
     [-rise] [-fall]
     [-min] [-max]
     resistance {port|port_bus}...
```

Sets the drive resistance on the specified input and/or bidirectional ports of the top cell.

The drive resistance models the cell driving the port. This resistance is used to compute the load dependent delay and the slew at the specified ports. The load-dependent delay is added to the arrival times.

In absence of these specifications, by default, all inputs are assumed to have an infinite drive strength. The default condition implies that the transition time at the input pins is 0.

The smaller the drive resistance value, the higher the drive strength. A resistance value 0 implies an infinite drive strength. As the drive value (resistance) increases, drive strength decreases and transition time increases.

The load-dependent delay and the slew are computed by multiplying the drive resistance by the wire capacitance of the net connected to the port.

The resolution is 1/1000. Genus ignores (does not use) the minimum values but the they can be passed to downstream tools through `write_sdc`. As a result, the tool sets the `external_resistance` attribute on the specified ports and removes the external driver (sets the `external_driver` attribute to NULL) on the specified ports.

Use the `report port` command to view drive resistances asserted on ports.

Options and Arguments

`-fall`	Specifies that the drive resistance is to be used for the falling transition at the specified ports. By default, if neither the `-rise` nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.
`-min`	Specifies that the drive resistance is to be used to compute the early path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-max`	Specifies that the drive resistance is to be used to compute the late path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.
{*port* | *port_bus*}	Specifies a list of input or bidirectional ports on which the drive resistance is set.
resistance	Specifies the value of the drive resistance. This value must be greater than or equal to zero. The drive resistance is in units found in the target technology library.
`-rise`	Specifies that the drive resistance is to be used for the rising transitions at the specified ports. By default, if neither the `-rise` nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.

Examples

- The following command sets the resistance for the minimum and maximum rise and fall transitions of the external driver of port `in` to 0.560.

```
legacy_genus:/> set_drive 0.56 [find / -port in]
legacy_genus:/> get_attr external_resistance [find / -port in]
0.560 0.560 0.560 0.560
```

- The following commands set the resistance for the minimum rise and maximum rise transition of the external driver of port fS_opb[1], respectively to 0.560 and 0.280.

```
legacy_genus:/designs/cpu_top/> set_drive 0.56 [find / -port S_opb[1]] -min
-ris
legacy_genus:/designs/cpu_top/> get_attr external_resistance [find / -port
S_opb[1]]
0.560 no_value no_value no_value
legacy_genus:/designs/cpu_top/> set_drive 0.28 [find / -port S_opb[1]] -max
-ris
legacy_genus:/designs/cpu_top/> get_attr external_resistance [find / -port
S_opb[1]]
0.560 no_value 0.280 no_value
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related attributes:	external_resistance
	external_driver

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_driving_cell

```
set_driving_cell
     {-cell | -lib_cell} cell [-library library]
     [-from_pin pin] [-pin pin]
     [-input_transition_rise float]
     [-input_transition_fall float]
     [-rise] [-fall] [-max | -min]
     [-multiply_by integer] [-no_design_rule]
     [-dont_scale]
     [-clock clock_list] [-clock_fall] port_list
```

Models the drive capability of an external driver connected to the input port.

To model the drive capability of the external driver, the timing analyzer computes an offset to the arrival time of an input, and changes the slew time used to compute the delay for the next cell. The timing analyzer computes the offset by subtracting the zero load delay from the total delay of the cell.

This constraint overrides any prior `set_drive` or `set_input_transition` constraint defined on the same port. If you do not specify drive or transition using any of these constraints, the timing analyzer use a default value of 0ps.

This command associates an external driving cell to the input or inout port of the current design.

Note: You cannot assign a buffer to drive the output port.

Options and Arguments

`-cell cell`
Specifies the name of the libcell used to drive the port.

`-clock`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-clock_fall`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-dont_scale`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-fall`

Indicates that the information applies only to a falling transition on the ports.

By default, if both the `-fall` and `-rise` options are omitted, the information applies to the rising and falling transitions on the ports.

`float`

Specifies the transition time for the specified port(s).

`-from_pin` *`pin`*

Specifies an input pin on the specified cell. The command uses the drive of the timing arc from this pin to the pin specified with the `-pin` option.

If this option is omitted, the transition on `-pin` is assumed to be 0.

`-input_transition_fall` *`float`*

Specifies the input falling transition time associated with the `-from_pin` option.

Default: `0`

`-input_transition_rise` *`float`*

Specifies the input rising transition time associated with the `-from_pin` option.

Default: `0`

`-lib_cell` *`cell`*

Specifies the name of the libcell used to drive the port.

`-library` *`library`*

Specifies the name of the library to which the libcell belongs.

If this option is omitted, the command searches in all available libraries.

`-max`

Indicates that the driver cell information applies to setup analysis.

By default, if both the `-max` and `-min` options are omitted, the information applies to setup analysis.

`-min`

Specifies the early drive assertion.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-multiply_by` *`integer`*

Specifies the total number of external drivers. The number of additional external drivers is the specified number minus 1 and this value will be set on the `external_non_tristate_drivers` port attribute.

Default: `1`

`-no_design_rule` Indicates to ignore the design rule violations seen by the external driver pin.

`-pin` *`pin`* Specifies the output pin of the libcell which will be used to drive the port.

If this option is omitted, the command chooses an output pin at random.

`port_list` Specifies the port(s) to which the external driver applies.

`-rise` Indicates that the information applies only to a rising transition on the ports.

By default, if both the `-fall` and `-rise` options are omitted, the information applies to the rising and falling transitions on the ports.

Examples

- The following command specifies `nand2` as driving cell for port `a/foo[3]`. The from_pin A of the `nand2` libcell has a rising input transition time of `0.1`.

```
set_driving_cell -lib_cell nand2 -from_pin A -input_transition_rise 0.10
[get_ports {a/foo[3]}]
```

- The following command sets the driving cell for all `-rise`, `-fall`, `-min`, and `-max` combinations:

```
set_driving_cell -lib_cell nand2 -library tutorial -pin "Y" [get_ports {a/
foo[3]}]
```

- The following command overrides the driving cell setting for only the `-rise` and `-min` options:

```
set_driving_cell -lib_cell inv1 -from_pin "A" -rise -min [get_ports {a/foo[3]}]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related attribute:	external_non_tristate_drivers
Related command:	reset_driving_cell on page 1498

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_equal

```
set_equal
     port1 port2
```

Declares the two specified ports in the current design to be logically equivalent. This information is used during synthesis to eliminate redundant ports and hence improving optimization quality.

Note: The command does not work on output ports and ports with multidriven nets.

Note: Use the reset_design command to remove this property from a port.

Options and Arguments

port1	Specifies the first input port.
port2	Specifies another input port.

Example

- The following command declares ports `in` and `in1` as logically equivalent.

```
legacy_genus:/> set_equal [find / -port in] [find / -port in1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_false_path

```
set_false_path
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {{clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|net|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|net|port_bus|pin_bus}...
      |-fall_through {port|instance|pin|net|port_bus|pin_bus}...}...
     [-rise] [-fall] [-setup] [-hold]
     [-exception_name name] [-reset_path]
     [-comment string]
```

Identifies false paths in a design, and breaks or disables specific instance timing arcs in a design resulting in them not being timed.

Note: To break combinational loops, use the `set_disable_timing` command instead of the `set_false_path` command.

The `set_false_path` command differs from the `set_disable_timing` command as follows:

- `set_false_path` command selectively disables the arrival time information depending on the precedence rules.
- `set_false_path` command does not affect constant values.
- `set_disable_timing` command physically snips a timing arc such that neither arrival times nor constant values propagate through it.
- `set_disable_timing` command is applied in all cases to valid timing arcs.

Th `set_false_path -from fpin` command disables all timing paths whose source pin is an `fpin`. Similarly, the `set_false_path -to tpin` command disables all timing paths whose sink pin is a `tpin`. Using the `set_false_path -from fpin -to tpin` command disables all timing paths whose source is fpin and sink is tpin.

Specify a group of pins by enclosing the group with curly braces ({}) within one `set_false_path` command.

To set a false path for any path beginning at `p1` or `p2` and terminating at `t1` or `t2`, group the pins as follows:

```
set_false_path -from {p1 p2} -to {t1 t2}
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: This method of pin grouping reduces the number of exceptions. Having large numbers of exceptions adversely impacts the run time of the timing analysis commands.

`set_clock_groups -asynchronous` also applies a `set_false_path` in between various clock groups, however, the interpretation of that command is different than `set_false_path` for implementation tools as `set_false_path` allows SI/EM analysis as if the clocks could be pulsed concurrently. Genus will still write out the original command used when `write_sdc` is involved to export the SDC constraints.

Options and Arguments

`-comment string`
: Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.

`-exception_name name`
: Specifies the name of the timing exception you want to create.

  Default: `dis_n`

`-fall`
: Applies the assertion at the falling edge of the signal on the path endpoints.

`(-from | -fall_from | -rise_from}`
`{clock | port | inst| hinst | pin | hpin | port_bus | hpin_bus}`
: Specifies a list of timing paths start points.

  The argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries.. The argument can be a collection.

  The valid start points are:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Input port
- Bidirectional port
- Clock pin of sequential cell
- Clock pin of sequential cell is one of the following:
  - Structural clock pin that is part of latch or flip-flop.
  - Inferred clock pin in cells like macro cells.
- A pin is an inferred clock pin if it is associated with one of the following:
  - Reference end of setup check
  - Reference end of hold check
  - Reference end of removal check
  - Reference end of recovery check
  - Minimum period check
  - Minimum pulse width check
  - Pin associated with set_input_delay

Applies the false path constraint to all paths originating from the list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points.

By default, the `-from` option applies the constraint to both the rising and the falling edges.
`-rise_from`: Applies the false path constraint at the rising edge of the signal on the from pins.
`-fall_from`: Applies the false path constraint at the falling edge of the signal on the from pins.

`-fall_through | -rise_through | -through`
`{`*`port | inst| hinst | pin | hpin | hnet | port_bus | hpin_bus`*`}`

Specifies a list of timing paths through points.

The argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The argument can be a collection.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Specify one of the following: `-through`:

When you use this parameter without using the `-from` or `-to` options, the false path constraint is set for all paths that go through the specified pins. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list and going through any point on the `-through` pin list and going to any point on the `-to` list is affected by the constraint.

By default, the `-through` option applies the constraint to both the rising and the falling edges.
`-rise_through`: Applies the assertion at the rising edge of the signal on the through pins.
`-fall_through`: Applies the assertion at the falling edge of the signal on the through pins.

`-fall_to | -rise_to | -to`
`{`*`clock`* `|` *`port`* `|` *`inst`*`|` *`hinst`* `|` *`pin`* `|` *`hpin`* `|` *`port_bus`* `|` *`hpin_bus`*`}`

Specifies a list of timing paths end points.

The argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The argument can be a collection.

The valid end points are:

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- Output port
- Bidirectional port
- Data pin of sequential cell
- Data pin of sequential cell is one of the following:
  - Structural data pin that is part of latch or flip-flop.
  - Inferred data pin in cells like macro cells.
- A pin is an inferred data pin if it is associated with one of the following:
  - Signal end of setup check
  - Signal end of hold check
  - Signal end of non-sequential setup check
  - Signal end of non-sequential hold check
  - Signal end of removal check
  - Signal end of recovery check
  - Pin associated with `set_external_delay`
  - Data pin of clock gating cell

Applies the false path constraint to all paths ending in the -to pin list. If a clock is specified as the -to object, all primary outputs and registers clocked by that clock are used as endpoint.

By default, the `-to` option applies the constraint to both the rising and the falling edges.
`-rise_to`: Applies the false path constraint at the rising edge of the signal on the to pins.
`-fall_to`: Applies the false path constraint at the falling edge of the signal on the to pins.

`-hold | -setup`

Sets false paths for either setup or hold analysis.

Note: Genus does not perform hold analysis and optimization. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

	Note: If you do not specify -`setup` or -`hold`, the software sets false paths for both setup and hold analysis.
`-reset_path`	Resets any other exceptions (`multicycle` and `path_delay`) on the path specified using the `-from`, `-through`, `-to` options and applies a `set_false_path` exception for these paths.
`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`-setup`	Applies the false path constraint for setup analysis only.

Examples

- The following command applies a false path (disables the path for timing analysis) to all paths between `inst1/CK` and inst2/D.

```
set_false_path  -from inst1/CK  -to inst2/D -comment "my false paths"
```

- The following command applies a false path (disables the path for timing analysis) through all paths that go through `inst1/A`.

```
set_false_path through inst1/A
```

Related Information

Related attribute:	(exception) comment
Related commands:	set_disable_timing on page 1558
	set_max_delay on page 1598
	set_min_delay on page 1612
	set_multicycle_path on page 1621
	set_clock_groups on page 1536

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_fanout_load

`set_fanout_load` *`float port_list`*

Specifies the fanout load on a set of ports. Library instance pins may have fanout loads set in the technology library. Fanout does not have specific units so it is assumed that fanout as used by constraints is consistent with fanout in the technology libraries. If a pin or port has no fanout attribute, either from constraints or from the technology library, the fanout load is zero.

Note: The fanout load is used only for design rule checking; it does not affect timing.

This command sets the `external_fanout_load` attribute on the specified output and inout ports.

Options and Arguments

`float`	Specifies the external load for the port(s).
`port_list`	Specifies the ports to which the fanout load is applied.

Example

- The following command specifies an external load of 45.6 for port `out`.

```
legacy_genus:/> set_fanout_load 45.6 [find / -port out]
legacy_genus:/> get_attr external_fanout_load [find / -port out]
45.600
```

Related Information

Related commands:	report nets on page 661
	report port on page 668
	set_max_fanout on page 1604
Sets this attribute:	external_fanout_load

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_hierarchy_separator

```
set_hierarchy_separator
     character
```

Specifies the hierarchy delimiter character used in the SDC file. Preferably, choose your hierarchy separator to be a character, which is not used in the actual naming of any object in your design.

For example, your design does not use any escaped naming. In that case, " / " is a good hierarchy separator. When SDC refers to " a/b", there is no ambiguity. On the other hand, if your design has several objects, where " / " is used as a part of the escaped name itself, then using this same character as a hierarchy separator can cause ambiguity.

In such situations, the hierarchy separator might be changed to another character like @, using the following command:

```
set_hierarchy_separator @
```

With the hierarchy separator being set at @, the above ambiguity is resolved.Then, "a/b" means a design object a/b, where "/" is part of the name. If you want to refer to "b within a", you need to specify a@b.

Options and Arguments

character	Specifies the hierarchy delimiter character. *Default*: /

Example

- In the following example the hierarchy delimiter is set to #. The next command returns the full path to the instances that match the specified pattern `inst10#inst1`. Only one instance is returned in this case.

```
set_hierarchy_separator #
get_cells inst10#inst1
/designs/top/instances_hier/inst10/instances_comb/inst1
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_ideal_latency

```
set_ideal_latency [-min | -max] [-rise | -fall]
     latency_value pin|port...
```

Specifies the ideal latency to use for input or output leaf-cell pins and top-level ports that are part of an ideal network. These set a point delay on the specified object.

Options and Arguments

Indicates that the specified value is a falling ideal latency value.

`-fall`	Indicates that the specified value is a falling ideal latency value.
`latency_value`	Specifies the ideal latency time for the specified port(s).
`-max`	Indicates that the specified value is a maximum ideal latency value.
`-min`	Indicates that the specified value is a minmum ideal latency value.
`pin``|`*`port`*	Specifies the list of objects - instance pins or ports.
`-rise`	Indicates that the specified value is a rising ideal latency value.

Examples

- The following command specifies the ideal latency of 1.5 to the input pin A of cell `FE_OFC413_scan_se`:

  ```
  set_ideal_latency 1.5 FE_OFB413_scan_se/A
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_ideal_net

`set_ideal_net` *`net_list`*

Causes the specified nets to be treated as ideal and prevents the timing and optimization engines from optimizing these nets. The command sets the `ideal_driver` attribute on the leaf-level drivers of the specified nets to `true`.

Options and Arguments

`net_list`	Specifies the nets which must be treated as ideal nets.

Example

```
set_ideal_net opa_5[1] opa_5[2]
```

Related Information

Related command:	remove_ideal_net on page 1480
Sets this attribute:	ideal_driver

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_ideal_network

`set_ideal_network {pin | port | net}... [-no_propagate]`

Sets the `ideal_network` attribute on all leaf-level drivers of the specified object to `true`.

Options and Arguments

`{pin | port | net}`
: Specifies the object to which the command applies.

`-no_propagate`
: Only sets the `ideal_driver` attribute on the leaf-level drivers of the specified object to `true`. This way only the electrically connected nets are idealized.

Related Information

Related command:	remove_ideal_network on page 1481
Sets these attributes:	ideal_driver
	ideal_network

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_input_delay

```
set_input_delay
     [-clock clock]
     [-clock_fall | -clock_rise]
     [-level_sensitive]
     [-network_latency_included]
     [-source_latency_included]
     [-reference_pin {pin|port}...]
     [-rise] [-fall] [-max] [-min]
     [-add_delay] [-name name] [-group_path group]
     delay {port|pin|port_bus}...
```

Constrains input and inout ports, port busses, and pins (that are valid start points) within the design relative to a clock edge.

If you omit both the `-min` and `-max` options, the delay is assumed to apply for both setup and hold analysis.

If you omit both the `-rise` and `-fall` options, the delay applies for both the rising and falling edges of the input signal.

Note: This command sets the `external_delays` attribute on the specified pins or ports.

Options and Arguments

`-add_delay`	Specifies to add the specified input delay information for the specified pins or ports. Use this option to capture the delay information if multiple paths lead to an input pin or port that are relative to different clocks or clock edges. If you omit this option, the specified delay information overwrites all existing input delays for the specified pins or ports.
`-clock` *`clock`*	Specifies the reference clock. The input delay is defined relative to this clock. If no clocks are specified, the timing paths from the specified pins and ports will be unconstrained as captured in the `report timing` report.
`-clock_fall`	Specifies to use the falling edge of the reference clock as reference edge.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: `-clock_rise`

`-clock_rise` Specifies to use the rising edge of the reference clock as reference edge.

Default: `-clock_rise`

`delay` Specifies the input delay value. This is a floating number.

`-fall` Specifies that the delay is measured with respect to the falling edge of the input signal.

`-group_path` *`group`* Specifies the name of the group to which paths ending at the specified ports or pins must be added

`-level_sensitive` Specifies that the constraint comes from a level-sensitive latch.

By default, the constraint comes from an edge-triggered flip-flop.

`-max` Indicates that the specified delay applies for setup analysis.

`-min` Indicates that the specified delay applies for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-name` *`name`* Associates a name with the specified timing constraint.

`-network_latency_included`

Specifies that the input delay includes the clock network latency values. This implies that the tool does not need to consider the clock network latency values when optimizing this path.

Sets the `clock_network_latency_included` attribute to `true`.

`{`*`port`* | *`pin`* | *`port_bus`*`}`

Specifies the input and inout pins, ports, and port buses to which the specified input delay value applies.

`-reference_pin {`*`pin`* | *`port`*`}`

Adds source and network latency of the specified reference pin to the input delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the -clock parameter.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-rise`
Specifies that the delay is measured with respect to the rising edge of the input signal.

`-source_latency_included`
Specifies that the input delay includes the clock source latency values. This implies that the tool does not need to consider the clock source latency values when optimizing this path.

Sets the `clock_source_latency_included` attribute to `true`.

Example

The following commands set an input delay on input port `en` with respect to clocks `clk1` and `clk2`. The example shows the effect of using the `-add_delay` option. When the second `set_input_delay` is specified without the `-add_delay` option, the first input delay is removed. However, when the third `set_input_delay` is specified with the `-add_delay` option, the delay information is stored.

```
legacy_genus:/> set_input_delay -clock clk1 2 en
legacy_genus:/> get_attr external_delays [find / -port en]
/designs/test2/timing/external_delays/in_del
legacy_genus:/> set_input_delay -clock clk2 5 en
   Removing external delay 'in_del'.
Info    : Removed object. [TUI-58]
       : Removed external_delay '/designs/test2/timing/external_delays/in_del'.
legacy_genus:/> get_attr external_delays [find / -port en]
/designs/test2/timing/external_delays/in_del_1_1
legacy_genus:/> set_input_delay -add -clock clk1 2 en
legacy_genus:/> get_attr external_delays [find / -port en]
/designs/test2/timing/external_delays/in_del_1_1 /designs/test2/timing/
external_delays/in_del
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Sets these attributes:	clock_network_latency_included
	clock_source_latency_included
	external_delays

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_input_transition

```
set_input_transition
     {-min | -max} {-rise | -fall}
     [-clock_fall] [-clock clock_list]
     float port_list
```

Sets a specific transition time (slew) on the specified input and inout ports. You can specify the minimum rise, minimum fall, maximum rise, or maximum fall slew.

Sets the `fixed_slew` attribute and removes the value from the `external_driver` attribute on the specified ports.

The minimum values are not used for timing analysis, but they will be written out when you execute the `write_sdc` command.

Options and Arguments

`-clock`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-clock_fall`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-fall` Indicates that the specified value is a falling transition value.

`float` Specifies the transition time for the specified port(s).

`-max` Indicates that the specified value is a maximum transition value.

`-min` Indicates that the specified value is a minimum transition value.

`port_list` Specifies the port(s) to which the transition time applies.

`-rise` Indicates that the specified value is a rising transition value.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command sets the minimum rise slew for input port `in` to `0.6`.

```
legacy_genus:/> set_input_transition -rise -min 0.6 /designs/top/ports_in/in
legacy_genus:/> get_attr fixed_slew /designs/top/ports_in/in
600.0 no_value no_value no_value
```

Related Information

Sets these attributes:	external_driver
	fixed_slew

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_lib_pins

```
set_lib_pins
     [-max_capacitance float] [-min_capacitance float]
     [-max_fanout integer] [-min_fanout integer]
     [-max_transition float] [-min_transition float]
     libpin_list
```

Sets the `max_fanout`, `max_transition` and `max_capacitance` attributes on the specified `libpin`.

Options and Arguments

`-max_capacitance` *`float`*

Specifies the maximum capacitive load that can be driven by the specified libcell pins.

`-max_fanout` *`integer`*

Specifies the maximum fanout that can be driven by the specified libcell pins.

`-max_transition` *`float`*

Specifies the maximum transition time on the specified libcell pins

`-min_capacitance` *`float`*

Specifies the minimum capacitive load that can be driven by the specified lib_cell pins.

`-min_fanout` *`integer`*

Specifies the minimum fanout that can be driven by the specified lib_cell pins.

`-min_transition` *`float`*

Specifies the minimum transition time on the specified lib_cell pins

`libpin_list`

Specifies the libcell pins to which the specified constraints apply.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command specifies that the maximum fanout that can be driven by pin `Y` of libcell `nand2` is 10.

```
legacy_genus:/> set_lib_pins -max_fanout 10 nand2/Y
```

Related Information

Sets these attributes:	max_capacitance
	max_fanout
	max_transition
	min_capacitance
	min_fanout
	min_transition

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_load

```
set_load
     {[-pin_load] [-subtract_pin_load] [-wire_load] port...
     | net...
     | port_bus ...}
     [-min] [-max] capacitance
```

Sets the capacitance to the specified value on the ports and nets in the design. If you omit both the `-min` and `-max` options, the capacitance applies for both setup and hold analysis.

For ports, the asserted capacitance is either the external pin capacitance when the `-pin_load` option is specified, or the external wire capacitance when the `-wire_load` option is specified, or the total external capacitance when neither option is specified.

Options and Arguments

capacitance
: Specifies the capacitance. This is a floating number.

`-max`
: Indicates that the capacitance applies for setup analysis.

`-min`
: Indicates that the capacitance applies for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-pin_load`
: Specifies the load of pins connected to this port. Sets the `external_pin_cap` attribute on the port.

Note: This option applies only to ports.

{*port* | *net* | *port_bus*}
: Specifies the port, net, or port_bus objects to which the specified capacitance applies.

`-subtract_pin_load`
: Subtracts the capacitance of the pins connected to the net of the port from the specified capacitance value before it applies it to the ports.

Note: This option applies only to ports.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-wire_load`	Specifies the load of the wire connected to this port. Sets the `external_wire_cap` attribute on the port. **Note:** This option applies only to ports.

Examples

- The following command sets an external minimum pin capacitance of 1.5 units on the in1in2 ports:

```
set_load 1.5 -min -pin_load [get_ports {in1 in2}]
```

- The following command sets an external minimum wire capacitance of 1.2 units on the in1in2 ports:

```
set_load 1.2 -min -wire_load [get_ports {in1 in2}]
```

- The following command sets an external maximum pin capacitance of 2.6 units on the in1in2 ports:

```
set_load 2.6 -max -pin_load [get_ports {in1 in2}]
```

- The following command sets an external maximum wire capacitance of 3.0 units on the in1in2 ports:

```
set_load 3.0 -max -wire_load [get_ports {in1 in2}]
```

Related Information

Sets these attributes:	external_pin_cap
	external_wire_cap
Related commands:	all_outputs on page 1403
	report port on page 668
	set_drive on page 1563
	set_resistance on page 1636

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_load_unit

```
set_load_unit
     {-picofarads | -femtofarads } [float]
```

Specifies the load unit used inside the SDC file or as part of the sdc command. By default, the time and load units use the read_sdc command's units whereas when using SDC commands interactively, the units of the first technology library are used.

Note: Although supported by the `read_sdc` command, it is *not* an official SDC command.

Options and Arguments

float	Indicates the number that any load value in subsequent SDC commands should be multiplied with.
`-femtofarads`	Indicates that the load values are specified in femto farads.
`-picofarads`	Indicates that the load values are specified in pico farads.

Examples

- The following command indicates that each load unit in the SDC file equals 5 femtofarads.

```
set_load_unit -femtofarads 5.0
set_load_unit -picofarads 1.0
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_logic_dc

```
set_logic_dc
     {pin | port} ...
```

If the specified object is an output port or an input pin, connects the object to constant 0.

If the specified object is an input port or an output pin, connects the loads of these objects to constant 0.

Note: This command does not work on pins or ports with multi-driver nets.

Options and Arguments

pin	Specifies the name of the pin(s) to which the command applies.
port	Specifies the name of the port(s) to which the command applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_logic_one

```
set_logic_one
     {pin | port} ...
```

Sets ports or pins to logic 1 in the design.

If the specified object is an output port or an input pin, connects the object to constant 1.

If the specified object is an input port or an output pin, connects the loads of these objects to constant 1.

Note: This command does not work on pins or ports with multi-driver nets.

Options and Arguments

pin	Specifies the name of the pin(s) to which the command applies.
port	Specifies the name of the port(s) to which the command applies.

Examples

- The following command sets the specified port to logic 1:

```
set_logic_one [get_ports {din[1]}]
```

Related Information

Related command: report case_analysis on page 589

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_logic_zero

```
set_logic_zero
     {pin | port} ...
```

Sets ports or pins to logic 0 in the design.

If the specified object is an output port or an input pin, connects the object to constant 0.

If the specified object is an input port or an output pin, connects the loads of these objects to constant 0.

Note: This command does not work on pins or ports with multi-driver nets.

Options and Arguments

pin	Specifies the name of the pin(s) to which the command applies.
port	Specifies the name of the port(s) to which the command applies.

Related Information

Related command: report case_analysis on page 589

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_capacitance

```
set_max_capacitance
     [-rise] [-fall]
     [-clock_path] [-data_path]
     float {design | port | clock | libpin}...
```

Specifies the maximum capacitance that can be driven by all ports in the specified design(s) or that can be driven by the specified ports.

Sets the `max_capacitance` attribute on the specified design(s) or port(s).

Options and Arguments

clock	Specifies the name of the clock to which the maximum capacitance design rule constraint applies.
`-clock_path`	Specifies to apply this design rule constraint to all pins that are in the network of the specific clocks in the object list. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-data_path`	Specifies to apply this design rule constraint to all pins that are in the data paths launched by the specific clocks in the object list. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
design	Specifies the name of the design to which the maximum capacitance design rule constraint applies.
`-fall`	Indicates that the specified value applies to the falling capacitance. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
float	Specifies the maximum capacitance value.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

libpin	Specifies the name of the library cell pin to which the maximum capacitance design rule constraint applies.
port	Specifies the names of the ports to which the maximum capacitance design rule constraint applies.
`-rise`	Indicates that the specified value applies to the rising capacitance. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Example

- The following command constrains the maximum capacitance that can be driven by port `in_port_1` to `0.5`.

```
set_max_capacitance 0.5 [get_ports in_port_1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_delay

```
set_max_delay
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|net|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|net|port_bus|pin_bus}...
      |-fall_through {port|instance|pin|net|port_bus|pin_bus}...}...
     [-rise] [-fall]
     [-exception_name name] [-comment string]
     [-combinational_from_to] [-reset_path] float
```

Specifies a maximum delay target for a timing path. The `set_max_delay` command takes precedence over other path exceptions specified for the same path, except for the set_false_path command.

Use either the -`from`, -`to`, or -`through` option with the `set_max_delay` command to identify the paths. Specifying both the -`from` and -`to` options identifies a path with specific starting and ending points.

When you specify a sequential cell name for starting point of the timing path, the valid path starts from the `clk` pin. This means the latency and delay from `clk` to output Q is also included in the delay. For path segmentation, you can specify the path to start from Q output of the register by adding /Q after the cell name. In this case, path from `clk` to Q is not considered in the calculation.

Note: Use only one -`from`, -`rise_from`, or -`fall_from`, -`to`, -`rise_to`, or -`fall_to` parameter per command. You can use multiple -`through` parameters in the same command.

Options and Arguments

`-combinational_from_to`

Defines a path defined by `-from` and `-to` pins as combinational.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-comment string`

Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.

`-exception_name name`

Specifies the name of the timing exception you want to create.

Default: `del_n`

`-fall`

Apply the assertion at the falling edge of the signal on the path endpoints.

`-fall_from | -rise_from {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Specifies a list of timing paths start points. The argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries.

The argument can be a collection.

The valid start points are:

- Input port
- Bidirectional port
- Clock pin of sequential cell
- Clock pin of sequential cell is one of the following:
- Structural clock pin that is part of latch or flip-flop.
- Inferred clock pin in cells like macro cells.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

A pin is an inferred clock pin if it is associated with one of the following:

- Reference end of setup check
- Reference end of hold check
- Reference end of removal check
- Reference end of recovery check
- Minimum period check
- Minimum pulse width check
- Pin associated with set_input_delay

-`rise_from`: Applies the maximum delay constraint at the rising edge of the signal on the from pins.
-`fall_from`: Applies the maximum delay constraint at the falling edge of the signal on the from pins.

`-from {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Applies the maximum delay constraint to all paths originating from the from argument list. If a clock is specified as the -`from` object, all primary inputs and registers clocked by that clock are used as start points.

By default the -`from` option applies the constraint to both the rising and the falling edges.

`-fall_through {port|inst|hinst|pin|hpin|hnet|port_bus|hpin_bus}`

Selects all paths whose signals have a falling edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-fall_to {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths whose signals have a falling edge at the specified end points.The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`delay_value` Specifies the maximum delay value.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-reset_path`
: Resets any other exceptions (`false_path` and `multicycle`) on the path specified using the `-from`, `-through`, `-to` options and applies a `set_max_delay` exception for these paths.

`-rise`
: Applies the assertion at the rising edge of the signal on the path endpoints.

`-rise_from {`*`clock`*`|`*`port`*`|`*`inst`*`|`*`hinst`*`|`*`pin`*`|`*`hpin`*`|`*`port_bus`*`|`*`hpin_bus`*`}`
: Selects all paths whose signals have a rising edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`{`*`port`*`|`*`inst`*`|`*`hinst`*`|`*`pin`*`|`*`hpin`*`|`*`hnet`*`|`*`port_bus`*`|`*`hpin_bus`*`}`
: Selects all paths whose signals have a rising edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-rise_to {`*`clock`*`|`*`port`*`|`*`inst`*`|`*`hinst`*`|`*`pin`*`|`*`hpin`*`|`*`port_bus`*`|`*`hpin_bus`*`}`
: Selects all paths whose signals have a rising edge on the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-through | -rise_through | -fall_through`
`{`*`port`*`|`*`inst`*`|`*`hinst`*`|`*`pin`*`|`*`hpin`*`|`*`hnet`*`|`*`port_bus`*`|`*`hpin_bus`*`}`
: Specifies a list of timing paths through points.

The argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries.

The argument can be a collection.

When you use this parameter without using the `-from` or `-to` options, all paths that go through the specified pins have the specified delay.

When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list and going through any point on the `-through` pin list and going to any point on the `-to` list is affected by the constraint.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

By default the `-through` option applies the constraint to both the rising and the falling edges.

`-rise_through`: Applies the assertion at the rising edge of the signal on the through pins.

`-fall_through`: Applies the assertion at the falling edge of the signal on the through pins.

`-to {clock|port|inst|hinst|pin|hpin|port_bus|hpin_bus}`

Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

Example

- The following command specifies a maximum delay of 10 time units to the path that starts from `inst1/CK` and ends at `inst2/D`.

```
set_max_delay  -from inst1/CK  -to inst2/D 10 -comment "my max delay"
```

- The following command specifies a maximum delay of 0.3 time units to the output port that has `output_delay` value also and the timing report along with the exception is shown below:

```
set_max_delay 0.3 -to port_pad_data_out[9]
/designs/dtmf_recvr_core/timing/exceptions/path_delays/del_1
```

```
report_timing -to [filter direction out[find / -port *]] -encounter
```

```
Path 1: VIOLATED Late External Delay Assertion
Endpoint:      port_pad_data_out[9]                                    (v)
Beginpoint:    TDSP_CORE_INST/DATA_BUS_MACH_INST_data_out_reg[9]/Q     (v)
```

```
Other End Arrival Time          0
- External Delay                300
+ Path Delay                    300
= Required Time                 0
- Arrival Time                  899
= Slack Time                   -899
```

Related Information

Related attribute: (exception) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_dynamic_power

```
set_max_dynamic_power
     float unit [design]
```

Specifies the maximum dynamic power constraint for the design and sets the `max_dynamic_power` design-level attribute. When a maximum dynamic constraint is set, Genus performs timing and dynamic power optimization simultaneously at all optimization stages like global mapping, refine mapping and incremental mapping. During incremental optimization, the value of the constraint is considered, while the constraint value is not taken into account during mapping. During global and refine mapping, logic on the non-critical paths is mapped to gates with less dynamic power dissipation. But during incremental mapping, the Genus-Low Power engine performs additional transformations including gate up-sizing or down-sizing, buffer removal and logic restructuring to further reduce the dynamic power dissipation of the design.

When both `max_leakage` and `max_dynamic` attributes are set, you must to set a weight factor using `lp_power_optimization_weight` design-attribute, so that Genus weighs the optimization cost function accordingly. If no weight factor is set, Genus optimizes only for the leakage power.

Options and Arguments

design	Specifies the name of the design to which the constraint applies.
float	Specifies the maximum dynamic power value.
unit	Specifies the power units. Following units can be specified: `pW`, `nW`, and `uW`.

Example

- The following command sets the maximum dynamic power constraint to 0.5 pW.

```
set_max_dynamic_power 0.5pW
```

Related Information

Sets this attribute:	max_dynamic_power
Related command:	set_max_leakage_power on page 1605

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_fanout

```
set_max_fanout
     float {design | port | libpin}...
```

Sets the maximum fanout load limit constraints on the specified instance pin, ports of the top cell, the specified modules, and/or clock waveforms, or lib pins. This constraint can be specified in SDC. When the net name is not specified in the constraint, then the constraints will be applied to all the nets in the design.

Sets the `max_fanout` attribute on the specified design(s) or port(s).

When set on a port, the fanout load of the net connected to the port must be less than or equal to the specified limit value to meet the constraint. The fanout load of a net is the sum of the fanout loads of the pins driven by the net. When set on a module, the limit applies to all the nets of that module. Libraries and library cell pins may also have the maximum fanout load limit. In this case the most constraining limit is applied to the nets.

Fanout load limit can be asserted on input ports, output pins, bidirectional pins, or ports and library pins. You cannot apply the constraint on input pins and output ports.

Options and Arguments

design	Specifies the name of the design to which the maximum fanout design rule constraint applies.
float	Specifies the maximum fanout value.
libpin	Specifies the name of the library cell pinto which the maximum fanout design rule constraint applies.
port	Specifies the names of the input ports to which the maximum fanout design rule constraint applies.

Example

The following command constrains the maximum fanout that can be driven by port `in_port_1` to `2.0`.

```
set_max_fanout 2.0 [get_ports in_port_1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_leakage_power

```
set_max_leakage_power
     float unit [design]
```

Specifies the maximum leakage power constraint for the design and sets the `max_leakage_power` design-attribute. When the leakage power constraint is set, Genus performs timing, area, and leakage power optimizations simultaneously during mapping without taking the constraint value into account. During global incremental and incremental optimization, the value of the constraint is considered.

When both `max_leakage` and `max_dynamic` attributes are set, you have to set a weight factor using `lp_power_optimization_weight` design-attribute so that Genus weighs the optimization cost function accordingly. If no weight factor is set, Genus optimizes only for the leakage power.

Options and Arguments

design	Specifies the name of the design to which the constraint applies.
float	Specifies the maximum leakage power value.
unit	Specifies the power units. Following units can be specified: `pW`, `nW`, and `uW`.

Example

- The following command sets the maximum leakage power constraint to 4pW.

```
legacy_genus:/> set_max_leakage_power 4pW
legacy_genus:/> get_attribute max_leakage_power /designs/top
0.004
legacy_genus:/> get_attribute lp_power_unit /
nW
```

Related Information

Sets this attribute:	max_leakage_power
Related command:	set_max_dynamic_power on page 1603

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_time_borrow

```
set_max_time_borrow
     float [clock | instance | pin | port ]...
```

Specifies the maximum time that can be borrowed by one stage from the next logic stage following a latch to meet timing constraints. Time borrowing is sometimes called cycle stealing. Time borrowing is performed automatically when performing timing analysis.

Sets the `latch_max_borrow` attribute.

Important

If the `latch_borrow` attribute has already been set on the specified object then the setting from this command is ignored.

Use the `set_max_time_borrow` assertion on pins or waveforms. If the assertion is placed on a clock waveform, all latches triggered by this clock signal get the specified maximum time borrow limit. If both the clock waveform arriving at the clock pin of a latch and the clock pin itself have time borrow limit constraints, the maximum borrow limit on the latch is decided by the assertion on the latch clock pin.

Options and Arguments

`{clock | instance | pin | port}`

Specifies the object(s) to which the borrow value applies.

You can set the borrow value for latch instances, clocks, latch enable pins, data pins and top-level ports.

If the command is set on multiple objects that overlap each other, for example a latch instance and its data pin, the minimum value will be taken.

float

Specifies the borrow value to be applied.

Example

- The following command sets a maximum borrow time of `0.5` for all latches clocked by `clk1`.

```
set_max_time_borrow 0.5 [find / -clock clk1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Sets this attribute: latch_max_borrow

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_max_transition

```
set_max_transition
     [-clock_path] [-data_path]
     [-rise] [-fall] float [design | clock | port | libpin]...
```

Specifies the maximum transition design rule for the specified design or port list.

Options and Arguments

`-clock_path`	Applies the max transition constraint only to pins in the clock network of the clocks defined in the object list.
`clock`	Specifies the name of the clock to which the maximum transition design rule constraint applies.
`-data_path`	Applies the max transition constraint only to data path pins, where the path is launched by the clocks defined in the object list.
`design`	Specifies the name of the design to which the maximum transition design rule constraint applies.
`-fall`	Applies the max transition constraint only to falling transitions. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`float`	Specifies the maximum transition. **Note:** The units are either the units set with the `set_time_unit` command or the units of the first technology library read in.
`libpin`	Specifies the name of the library cell pin to which the maximum transition design rule constraint applies.
`-rise`	Applies the max transition constraint only to rising transitions. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`port`	Specifies the names of the ports to which the maximum transition design rule constraint applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following command sets the maximum transition on the design to 4.

```
legacy_genus:/designs/rct> set_max_transition 4
legacy_genus:/designs/rct> get_attr max_transition
4000.0
```

In this example, the time unit that applied when the command was given was nanoseconds, while in the tool the value is stored in picoseconds.

Related Information

Sets these attributes:	(design) max_transition
	(port) max_transition

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_min_capacitance

```
set_min_capacitance
     [-rise] [-fall]
     [-clock_path] [-data_path]
     float {design | port | clock | libpin}...
```

Specifies the minimum capacitance that can be driven by all ports in the specified design(s) or that can be driven by the specified ports.

Note: Genus ignores this command for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the command out for consumption by downstream tools.

Options and Arguments

clock	Specifies the name of the clock to which this design rule constraint applies.
`-clock_path`	Specifies to apply this design rule constraint to all pins that are in the network of the specific clocks in the object list.
`-data_path`	Specifies to apply this design rule constraint to all pins that are in the data paths launched by the specific clocks in the object list.
design	Specifies the name of the design to which this design rule constraint applies.
`-fall`	Indicates that the specified value applies to the falling capacitance.
float	Specifies the minimum capacitance value.
libpin	Specifies the name of the library cell pin to which the minimum capacitance design rule constraint applies.
port	Specifies the names of the ports to which the minimum capacitance design rule constraint applies.
`-rise`	Indicates that the specified value applies to the rising capacitance.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

- The following command constrains the minimum capacitance that can be driven by port `in_port_1` to `0.5`.

```
set_min_capacitance 0.5 [get_ports in_port_1]
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_min_delay

```
set_min_delay
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|net|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|net|port_bus|pin_bus}..
      |-fall_through {port|instance|pin|net|port_bus|pin_bus}...}...
     [-rise] [-fall]
     [-exception_name name] [-comment string]
     [-combinational_from_to] [-reset_path] float
```

Constrains the specified timing paths by the given delay value for hold analysis. This value is not used for timing analysis by Genus. But it will be written out using `write_sdc`.

Options and Arguments

`-combinational_from_to`

Defines a path defined by `-from` and `-to` pins as combinational.

`-comment string`

Specifies a comment to be tagged to this command.

`-exception_name`

Specifies the name of the timing exception you want to create.

Default: `del_n`

`-fall`

Limits the path selection to those paths who have a falling edge at the end points.

`-fall_from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-fall_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge at the specified end points.The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

float

Specifies the delay constraint value, in picoseconds, by which the path has to be constrained for hold analysis.

`-from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that start from the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-reset_path`

Resets any other exceptions (`false_path` and `multicycle`) on the path specified using the `-from`, `-through`, `-to` options and applies a `set_max_delay` exception for these paths.

`-rise`

Limits the path selection to those paths who have a rising edge at the path end points.

`-rise_from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-rise_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-rise_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

`-through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths that traverse through the specified points. The points to traverse can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

Example

- The following command constrains the specified timing paths from port `in` by 0.6 time units for hold analysis.

```
legacy_genus:/> set_min_delay -from [find / -port in] 0.6
/designs/topmost/timing/exceptions/path_delays/del_2 -comment "my min delay"
```

Related Information

Related attribute: (exception) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_min_fanout

```
set_min_fanout
     float {design | port | libpin}...
```

Specifies the minimum fanout that can be driven by all input ports in the specified design(s) or that can be driven by the specified input ports.

Note: Genus ignores this command for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the command out for consumption by downstream tools.

Options and Arguments

design	Specifies the name of the design to which the minimum fanout design rule constraint applies.
integer	Specifies the minimum fanout value.
libpin	Specifies the name of the library cell pin to which the minimum fanout design rule constraint applies.
port	Specifies the names of the input ports to which the minimum fanout design rule constraint applies.

Example

- The following command constrains the minimum fanout that can be driven by port `in_port_1` to `2.0`.

  ```
  set_min_fanout 2.0 [get_ports in_port_1]
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_min_pulse_width

```
set_min_pulse_width
     [-low] [-high] {pin | port | clock} ... float
```

Selects the most restrictive value when multiple pulse width checks are set.

Note: This constraint is not used for timing analysis by Genus. The command will be read with `read_sdc` and will be written out with `write_sdc`.

Options and Arguments

float	Specifies the non negative floating minimum pulse width value. No default value is assumed for minimum pulse width checks.
`-high`	Applies the maximum pulse width check to high signal levels only.
`-low`	Applies the minimum pulse width check to low signal levels only. **Note:** If you specify neither the `-low` or the `-high` option, the constraint applies to both the low and high pulses of the clock signals
{pin | hpin | port | clock}	Specifies the list of pins, ports or clocks for which the minimum pulse width constraint must be checked.

Examples

- The following command sets a minimum pulse width, both low and high, check of 1.0 to the clock network of the current design:

  ```
  set_min_pulse_width 1.0
  ```

- The following command sets a minimum pulse width low check of 2.0 to all the pins clocked by the CLK1 clock:

  ```
  set_min_pulse_width -low 2.0 [get_clocks CLK1]
  ```

Related Information

Related commands: report min_pulse_width on page 647

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

report timing on page 709

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_min_transition

```
set_min_transition
     [-clock_path] [-data_path]
     [-rise] [-fall] float [design | clock | port | lib_pin]...
```

Specifies the minimum transition design rule for the specified design or port list.

Note: Genus ignores this command for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the command out for consumption by downstream tools.

Options and Arguments

`-clock_path`	Applies the minimum transition constraint only to pins in the clock network of the clocks defined in the object list.
`clock`	Specifies the name of the clock to which the minimum transition design rule constraint applies.
`-data_path`	Applies the minimum transition constraint only to data path pins, where the path is launched by the clocks defined in the object list.
`design`	Specifies the name of the design to which the minimum transition design rule constraint applies.
`-fall`	Applies the minimum transition constraint only to falling transitions.
`float`	Specifies the minimum transition. **Note:** The units are either the units set with the `set_time_unit` command or the units of the first technology library read in.
`lib_pin`	Specifies the name of the library cell pin to which the minimum transition design rule constraint applies.
`-rise`	Applies the minimum transition constraint only to rising transitions.
`port`	Specifies the names of the ports to which the minimum transition design rule constraint applies.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_mode

```
set_mode [-type cell]
     list_of modes
     object_list
```

Specifies the Liberty timing library modes to be active for a specific instance.

A library mode is a subset of active timing arcs that is defined in the timing library. Library modes are similar to conditional timing arcs, but do not require the application of constants to be set. Instead of changing a cell's mode by applying constants to its input pins, you can use this command to refer to sets of active arcs by name (such as READ or WRITE, for a memory cell).

Note: By default, all library modes in the timing library are active. If you specify one or more library modes using the `set_mode` command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive.

To reset the library modes for an instance, use the reset_mode command. Specify new value for the specified object attribute elements.

Note: Setting the library mode at the instance level is the only form of set_mode that is currently supported. This parameter exists to support constraints coming from external sources.

Options and Arguments

list_of_modes	Specifies the name of the library mode value defined in the timing library. You can specify one or more mode values, but only one mode value per defined mode group in the library.
object_list	Specifies the instance list to which to apply the library mode.
-type *cell*	Specifies the Liberty cell type of the instance.

Examples

- The following command specifies that only the timing arcs consistent with the `tst_mod_dsiable` library mode are active for instances of cell type `i_ram`. The `tst_mode_enable` library mode is made inactive.

  ```
  set_mode tst_mode_disable [get_cells i_ram]
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Related Information

Related command: reset_mode on page 1512

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_multicycle_path

```
set_multicycle_path
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|net|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|net|port_bus|pin_bus}...
      |-fall_through {port|instance|pin|net|port_bus|pin_bus}...}...
     [-rise] [-fall] [-setup] [-hold] [-start] [-end]
     [-exception_name name] float
     [-comment string] [-reset_path]
```

Creates a timing exception object that overrides the default clock edge relationship for paths that meet the path selection criteria. Multicycle paths are timing paths that require more than a single cycle for execution. By default, all paths are considered as single cycle paths for setup analysis, that is, data launched at any edge of the clock should be captured by the next flop at the next rising edge of the clock on to the destination flop.

Options and Arguments

`-comment` *`string`*
: Specifies a comment to be tagged to this command. The comment is retained by the tool and the same comment string is written out as part of the `set_multicycle_path` constraint when the constraints are dumped out using the `write_sdc` command.

`-end`
: Specifies whether the multi-cycle path constraint must be applied relative to the ending clock edge for the path.

  By default, the setup constraint is applied relative to the ending clock edge for the path.

`-exception_name` *`name`*
: Specifies the name of the timing exception you want to create.

  Default: `mc_`*`n`*

`-fall`
: Limits the path selection to those paths who have a falling edge at the end points.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall_from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-fall_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

-`fall_through` can be specified more than once in a single command.

`-fall_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge at the specified end points.The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

float

Specifies the number of cycles allowed for the path for setup or hold with respect to start point or end point clock.-`setup` value is used for setup path calculations.-`hold` value is used for hold path calculations.

If -`setup` or -`hold` is not specified, this value is used for `setup` and 0 is used for `hold`.

`-from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that start from the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

For example: If a clock is specified, all primary inputs and flops related to that clock will be used as path startpoints.

`-hold`

Applies the multi-cycle path constraint for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: setup

`-reset_path`

Resets any other exceptions (`false_path` and `path_delay`) on the path specified using the `-from`, and `-to` options and applies a `set_multicycle_path` exception for these paths.

- `-to` option will reset all paths pointing to the specified end points.
- `-from` option will reset all paths leading from the specified start points.
- If `-from` and `-to` are used together, only paths between these points will be reset.

`-rise`

Limits the path selection to those paths who have a rising edge at the path end points.

`-rise_from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-rise_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-rise_through` can be specified more than once in a single command.

`-rise_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

`-setup`

Applies the multi-cycle path constraint for setup analysis.

Default: setup

`-start`

Specifies whether the multi-cycle path constraint must be applied relative to the starting clock edge for the path.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

By default, the hold constraint is applied relative to the starting clock edge for the path.

`-through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths that traverse through the specified points. The points to traverse can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list of start points.

`-to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

Example

- The following command specifies that all paths from instance `ff1` to instance `ff2` require 2 cycles for execution for setup analysis.

```
set_multicycle_path -setup 2 -from ff1 -to ff2 -comment "my multicycle path"
```

Related Information

Related attribute: (exception) comment

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_operating_conditions

```
set_operating_conditions
     [-analysis_type string] [-object_list string]
     { -max_library string [-min_library string ]
     | -library string }
     {-min string -max string |operating_condition}
```

Specifies the operating conditions to be used for timing. Sets the `operating_conditions` attribute.

Options and Arguments

`-analysis_type` *`string`*

The only accepted value is `single`. This indicates that only one operating condition is to be used.

`-library` *`string`*

Specifies the library containing operating condition definitions.

`-max` *`string`*

Specifies the operating condition for setup analysis.

`-max_library` *`string`*

Specifies the library under which the operating condition is present. This operating condition is used for setup analysis.

`-min` *`string`*

Specifies the operating condition for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-min_library` *`string`*

Specifies the library under which the operating condition is present for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-object_list`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`operating_condition`

Specifies the operating conditions to be used for timing analysis.

Example

- The following command specifies to use operating condition `typical_case` from library `tutorial` for timing analysis.

```
set_operating_conditions -library tutorial typical_case
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_opposite

```
set_opposite
     port port
```

Declares the two specified ports to be logically opposite.

Note: The command does not work on output ports and ports with multidriven nets.

Example

- The following command declares ports `in` and `in1` as logically opposite.

  ```
  legacy_genus:/> set_opposite [find / -port in] [find / -port in1]
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_output_delay

```
set_output_delay
     [-clock clock]
     [-clock_fall] [-clock_rise]
     [-level_sensitive]
     [-network_latency_included]
     [-source_latency_included]
     [-reference_pin pin|port ...]
     [-rise] [-fall] [-max] [-min]
     [-add_delay] [-name name] [-group_path group]
     delay {port|pin|port_bus}...
```

Constrains output and inout ports, port busses, and pins (that are valid end points) within the design relative to a clock edge.

If you omit both the `-min` and `-max` options, the delay is assumed to apply for both setup and hold analysis.

If you omit both the `-rise` and `-fall` options, the delay applies for both the rising and falling edges of the output signal.

Note: This command also sets the `external_delays` attribute on the specified pins or ports.

Options and Arguments

`-add_delay`	Specifies to add the specified output delay information for the specified pins or ports. Use this option to capture the delay information if multiple paths lead to an output pin or port that are relative to different clocks or clock edges. If you omit this option, the specified delay information overwrites all existing output delays for the specified pins or ports.
`-clock` *`clock`*	Specifies the reference clock. The output delay is defined relative to this clock. If no clocks are specified, the timing paths to the specified pins and ports will be unconstrained as captured in the `report timing` report.
`-clock_fall`	Specifies to use the falling edge of the reference clock as reference edge.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Default: `-clock_rise`

`-clock_rise` Specifies to use the rising edge of the reference clock as reference edge.

Default: `-clock_rise`

`delay` Specifies the output delay value. This is a floating number.

`-fall` Specifies that the delay is measured with respect to the falling edge of the output signal.

`-group_path` *`group`* Specifies the name of the group to whichpaths starting at the specified ports or pins must be added.

`-level_sensitive` Specifies that the constraint goes to a level-sensitive latch.

By default, the constraint goes to an edge-triggered flip-flop.

`-max` Indicates that the specified delay applies for setup analysis.

`-min` Indicates that the specified delay applies for hold analysis.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-name` *`name`* Associates a name with the specified timing constraint.

`-network_latency_included`

Specifies that the output delay includes the clock network latency values. This implies that the tool does not need to consider the clock network latency values when optimizing this path.

Sets the `clock_network_latency_included` attribute to `true`.

{*`port`* | *`pin port_bus`*}

Specifies the output and inout pins, ports, and port buses to which the specified output delay value applies.

`-reference_pin` {*`pin`* | *`port`*}

Adds source and network latency of the specified reference pin to the output delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the -clock parameter.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-rise`
Specifies that the delay is measured with respect to the rising edge of the output signal.

`-source_latency_included`
Specifies that the output delay includes the clock source latency values. This implies that the tool does not need to consider the clock source latency values when optimizing this path.

Sets the `clock_source_latency_included` attribute to `true`.

Example

The following commands set an output delay on output port `gck` with respect to clocks `clk1` and `clk2`. The example shows the effect of using the `-add_delay` option. When the second `set_output_delay` is specified without the `-add_delay` option, the first output delay is removed. However, when the third `set_input_delay` is specified with the `-add_delay` option, the delay information is stored.

```
legacy_genus:/> set_output_delay -clock clk1 3 [find / -port gck]
legacy_genus:/> get_attr external_delays [find / -port gck]
/designs/test2/timing/external_delays/ou_del
legacy_genus:/> set_output_delay -clock clk2 5 [find / -port gck]
   Removing external delay 'ou_del'.
Info    : Removed object. [TUI-58]
       : Removed external_delay '/designs/test2/timing/external_delays/
ou_del'.
legacy_genus:/> get_attr external_delays  [find / -port gck]
/designs/test2/timing/external_delays/ou_del_1_1
legacy_genus:/> set_output_delay -clock clk1 3 [find / -port gck] -add_delay
legacy_genus:/> get_attr external_delays  [find / -port gck]
/designs/test2/timing/external_delays/ou_del_1_1 /designs/test2/timing/
external_delays/ou_del
```

Related Information

Sets these attributes: clock_network_latency_included

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

clock_source_latency_included

external_delays

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_path_adjust

```
set_path_adjust
     {{-from {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_from {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_from {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-to {clock|port|instance|pin|port_bus|pin_bus}...
      |-rise_to {clock|port|instance|pin|port_bus|pin_bus}...
      |-fall_to {clock|port|instance|pin|port_bus|pin_bus}...}
     |{-through {port|instance|pin|port_bus|pin_bus}...
      |-rise_through {port|instance|pin|port_bus|pin_bus}...
      |-fall_through {port|instance|pin|port_bus|pin_bus}...}...
     [-rise] [-fall]
     [-setup] [-hold]
     [-exception_name name] [-comment string]
     float'
```

Specifies the delay constraint value by which the path has to be adjusted. It calls the Genus native command `path_adjust`.

Path Adjust is a feature that allows users to specify the delay constraint value, overwriting the calculated value. The advantage of using path adjust is that it is not written out in the SDC constraints, so the place-and-route tool sees the timing.

Use the `set_path_adjust` command selectively to improve timing for certain paths in timing-critical designs because it might increase area.

Options and Arguments

-comment string	Specifies a comment to be tagged to this command.
-exception_name	Specifies the name of the timing exception you want to create. *Default*: `adj_n`
-fall	Limits the path selection to those paths who have a falling edge at the end points.
-fall_from {*clock*|*port*|*instance*|*pin*|*port_bus*|*pin_bus*}	Selects all paths whose signals have a falling edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-fall_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-fall_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a falling edge at the specified end points.The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

float

Specifies the delay constraint value, in SDC time units (instead of in picoseconds), by which the path has to be adjusted. A positive adjustment relaxes the clock constraint and a negative adjustment tightens it.

`-from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that start from the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

`-hold`

Specifies the adjust value for early paths.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Default: setup

`-rise`

Limits the path selection to those paths who have a rising edge at the path end points.

`-rise_from {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge at the specified start points. The start points can be input ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-rise_through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified through points. The through points can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-rise_to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths whose signals have a rising edge on the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

`-setup`

Applies the path adjust constraint for setup analysis.

Default: setup

`-through {port|instance|pin|net|port_bus|pin_bus}`

Selects all paths that traverse through the specified points. The points to traverse can be ports of your design, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances or nets. Specify a Tcl list.

`-to {clock|port|instance|pin|port_bus|pin_bus}`

Selects all paths that end at the specified end points. The end points can be output ports of your design, clock pins of flip-flops, clock objects, instances, or a combination of these. Specify a Tcl list of end points.

Example

- The following example relaxes the timing constraints on all timing paths starting from port `in` by 0.5 time units.

```
legacy_genus:/> set_path_adjust -from [find / -port in] 0.5
/designs/topmost/timing/exceptions/path_adjusts/adj_1
```

Related Information

Related command: path_adjust on page 475

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_port_fanout_number

`set_port_fanout_number` *`integer port_list`*

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.

The number of pins minus one is equal to the fanout of this port outside the design. This value is set on the `external_wireload_fanout` attribute.

Options and Arguments

`integer`	Specifies the number of external fanout pins. Since the value is the number of pins in the fanout of the port, it is expected to be an integer.
`port_list`	Specifies the ports for which the number is specified.

Example

- The following command specifies 5 fanout pins for port `out1`. The tool stores the value 4 (5-1) on the `external_wireload_fanout` port attribute for `out1`.

```
legacy_genus:/> set_port_fanout_number 5 out1
legacy_genus:/> get_attr external_wireload_fanout out1
4
```

Related Information

Sets this attribute: external_wireload_fanout

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_resistance

```
set_resistance
     net [net]...
     [-min] [-max] resistance
```

Sets the wire resistance to the specified value on the specified nets in the design. If you omit both the `-min` and `-max` options, the resistance is used as the minimum and maximum resistance.

Genus ignores this command for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the command out for consumption by downstream tools.

Options and Arguments

`-max`	Specifies the maximum resistance.
`-min`	Specifies the minimum resistance.
`net`	Specifies a list of nets on which the wire resistance must be set.
`resistance`	Specifies the value for the wire resistance. This value is in units found in the target technology library.

Related Information

Sets these attributes:	(fnet) resistance
	(net) resistance

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_size_only

```
set_size_only
     {subdesign|instance|net|design|libcell}...
```

Sets the `preserve` attribute for an object to `map_size_ok` if the cell is an unmapped sequential cell (unmapped combinational cells are ignored), and sets the `preserve` attribute to `size_ok` if the cell is mapped.

Note: The command `set_size_only` sets the attribute preserve to `map_size_ok` for a mapped instance only when the instance is totally unpreserved, that is when `preserve` = `FALSE`. Otherwise, it doesn't change the preserve attribute.

Options and Arguments

`{subdesign|instance|net|design|libcell}...`

Specifies the object(s) to be considered for sizing.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_timing_derate

```
set_timing_derate
     [-cell_delay] [-cell_check] [-net_delay]
     [-early] [-late] [-rise] [-fall]
     [-dynamic] [-static] [-clock] [-data]
     [-delay_corner delay_corner]
     [-power_domain power_domain][-mean][-sigma]
     float {libcell | instance | design}...
```

Models the on-chip variation (OCV) or delay change across a die due to variations in manufacturing process, voltage, and temperature that must be taken into account during timing analysis.

Note: Omitting both the `-cell_check` and `-cell_delay` options results in the derating factor being applied to both delay and constraint arcs.

Genus only supports some of the option combinations, but does store all information and writes it out with the `write_sdc` command.

Options and Arguments

`-cell_check`	Applies the derating factor to the timing checks.
`-cell_delay`	Applies the derating factor to the cell delays.
`-clock`	Applies the derating factors to clock paths only. **Note:** Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.
`-data`	Applies the derating factor to data paths.
`-delay_corner` *`delay_corner`*	Specifies the delay corner for which the modeling must be done.
`design`	Specifies the name of the top-level design to which the derating factor applies.
`-dynamic`	Applies the specified derating factor to the delta portion of net delays only.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-early`
: Applies the derating factor for early paths.

  Genus only supports this option in combination with the `-data` and `-cell_delay` options.

`-fall`
: Applies the value only to the fall delays.

`float`
: Specifies the derating value.

  Specify a positive derating value. An error will be given if you specify a negative derating value.

`instance`
: Specifies the instance(s) to which the derating factor applies.

  Genus does *not* support this option in combination with the `-clock` and `-cell_check` options.

`-mean`
: Specifies to apply separate derating value to the mean components of the delay.

`-late`
: Applies the derating factors for the late mode.

  Genus does *not* support this option in combination with the `-clock` option.

`libcell`
: Specifies the library cells to which the derating factor applies.

`-net_delay`
: Applies the derating factor to net delays.

  Note: This option is only available in the physical flows.

`-power_domain` *`power_domain`*
: Specifies the power domain for which the modeling must be done.

`-rise`
: Applies the value only to the rise delays.

`-sigma`
: Specifies to apply separate derating value to the sigma components of the delay.

`-static`
: Applies the specified derating factor to the non-delta portion of the net delays only.

  You can specify this option with the `-cell_delay` and `-net_delay` options.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Support Matrix

All option combinations are stored and written out during `write_sdc`.

Option combination			Combination supported?
`-early`	`-clock`	`-cell_delay`	Not supported.
`-early`	`-clock`	`-cell_check`	Not supported.
`-early`	`-data`	`-cell_delay`	Supported on instances and libcells for data-to-data checks.
`-early`	`-data`	`-cell_check`	Not supported.
`-late`	`-clock`	`-cell_delay`	Not supported.
`-late`	`-clock`	`-cell_check`	Not supported.
`-late`	`-data`	`-cell_delay`	Supported on instances and libcells.
`-late`	`-data`	`-cell_check`	Supported on libcells

- The following command scales the delay of libcell `DFFHQX1` by `2.3`.

  ```
  set_timing_derate -cell_delay 2.3 [find /lib*/* -libcell DFFHQX1]
  ```

  This command is equivalent to the following native command:

  ```
  set_attribute cell_delay_multiplier 2.3 [find /lib*/* -libcell DFFHQX1]
  ```

- The following command scales the constraint arcs of libcell `DFFHQX1` by `3.1`.

  ```
  set_timing_derate -cell_check 3.1 [find /lib*/* -libcell DFFHQX1]
  ```

  This command is equivalent to the following native command:

  ```
  set_attribute constraint_multiplier 3.1 [find /lib*/* -libcell DFFHQX1]
  ```

Related Information

Related command:	reset_timing_derate on page 1519
Sets these attributes:	(libcell) cell_delay_multiplier
	(inst) cell_delay_multiplier
	(libcell) cell_min_delay_multiplier
	(inst) cell_min_delay_multiplier

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_time_unit

```
set_time_unit
     {-picoseconds | -nanoseconds } [float]
```

Specifies the time unit used inside the SDC file or as part of the `dc::` command

Note: Although supported by the `read_sdc` command, it is *not* an official SDC command.

Options and Arguments

float	Indicates the number that any timing value in subsequent SDC commands should be multiplied with.
-nanoseconds	Indicates that the timing values are specified in nanoseconds.
-picoseconds	Indicates that the timing values are specified in picoseconds.

Examples

- The following command indicates that each timing unit in the SDC file equals 3 picoseconds.

  ```
  set_time_unit -picoseconds 3
  ```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_unconnected

```
set_unconnected
    port_list
```

Edits the netlist by disconnecting the output port.

Note: Although supported by the `read_sdc` command, it is *not* an official SDC command.

Options and Arguments

port_list	Specifies the names of the output ports that must be disconnected.

Related Information

Related command:	check_design on page 560

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_units

```
set_units
     [-capacitance string] [-resistance string]
     [-voltage string] [-current string]
     [-time string] [-power string]
```

Specifies the unit of capacitance, resistance, voltage, current, timing and power in SDC. This command only affects the unit in SDC, but not the units used in Genus calculations. For example, after `set_unit -time 1ns`, the timing unit used in Genus calculation is still `ps`. By default, the units used in SDC file is read from the first library.

Note: This command was added in SDC 1.7.

Options and Arguments

`-capacitance` *`string`*

Specifies the capacitance unit. String can have the following format:

`[`*`float`*`] {fF | pF | nF | uF | mF | kF | MF}`

`-current` *`string`*

Specifies the current unit.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-power` *`string`*

Specifies the power unit.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-resistance` *`string`*

Specifies the resistance unit.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-time` *`string`*

Specifies the time unit. String can have the following format:

`[`*`float`*`] {fs | ps | ns | us | ms | ks | Ms}`

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-voltage` *`string`* Specifies the voltage unit.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

Examples

- The following command specifies that one load unit is 0.5pF.

```
set_units capacitance 0.5fF
```

- The following command specifies that one time unit is 0.78ps.

```
set_units time 0.78ps
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_wire_load_mode

```
set_wire_load_mode {top | enclosed| segmented}
```

Sets the `wireload_mode` root attribute with the specified value.
The `enclosed` mode is the default mode. If the `default_wire_load_mode` is specified in the first technology library, then, that value is used.

Options and Arguments

`enclosed`	Uses the wire-load model of the smallest block that fully encloses the net to compute the load of the net. Hierarchical boundary pins are not counted as fanouts.
`segmented`	Divides nets that cross hierarchical boundaries into segments with one segment for each level of hierarchy. Separate load values are computed for each segment (counting the hierarchical boundary pins as individual fanouts) and the load values are added together.
`top`	Uses the wire-load model of the top-level design for all nets in all subdesigns. Hierarchical boundary pins are not counted as fanouts.

Example

```
legacy_genus:/> set_wire_load_mode top
```

Related Information

Sets this attribute:	wireload_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_wire_load_model

```
set_wire_load_model
     -name {auto_select | model_name | none | inherit}
     [-library library] [-min] [-max]
     {design | subdesign | instance | port}...
```

Forces Genus to use a specific wire load model for the given design, subdesigns, hierarchical instance or port.

Sets the `force_wireload` attribute if the specified object is a design, subdesign or a hierarchical instance. If the specified object is a hierarchical instance, the attribute on the corresponding subdesign is set.

Sets the `external_wireload_model` attribute if the specified object is a port.

Options and Arguments

design | *subdesign* | *instance*| *port*

Specifies the design, subdesign, hierarchical instance, or port for which you want to specify the wire load model.

If no object is specified, the wire load model is applied to the top-level design.

`-library` *library*

Specifies the technology library to which the wire load model belongs.

`-max`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-min`

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

`-name {auto_select | model_name | none | inherit}`

Specifies the name of the wire load model to use.

`auto_select` Automatically selects wire-load models according to the wire-load selection table or default wire-load model in the technology library.

Note: This value can only be applied for the design, subdesigns, or hierarchical instances.

`model_name` Forces Genus to use the specified custom wire-load model. Specify the hierarchical path to the wire-load model to be used.

`inherit` Causes the same behavior as `auto_select`.

Note: This value can only be applied for the design, subdesigns, or hierarchical instances.

`none` Prevents the use of any wire-load models.

Note: This value can only be applied for the design, subdesigns, or hierarchical instances.

Examples

- The following example forces the use of wire load model `10x10` for design `test_top`:

```
legacy_genus:/> set_wire_load_model -name [find /libraries -wireload "10x10"]\
[find / legacy_genus:/-design test_top]
```

Related Information

Sets these attributes:	force_wireload
	external_wireload_model
Related attributes:	wireload_mode
	wireload_selection

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

set_wire_load_selection_group

```
set_wire_load_selection_group
     [-library library] [-min] [-max]
     [-object_list string] name
```

Specifies a wire-load selection group.Sets the `wireload_selection` root attribute with the specified value.

Options and Arguments

`-library` *`library`*

Specifies the technology library to which the selection group belongs. If not, search all libraries.

If this option is omitted, the command searches for the specified wire-load selection table in the list of the libraries loaded. If the same table name appears in multiple libraries, the selection table from the earlier loaded library is used.

`-max`

Controls the choice of a wireload selection table only for the PVT max corner.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`-min`

Controls the choice of a wireload selection table only for the minimum PVT corner.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

`name`

Specifies the wire-load selection table to use. Specify the hierarchical path to the wire-load selection table to be used, if multiple selection tables with the same name exist.

`-object_list`

Specifies the object list on which the selection table applies.

Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The `write_sdc` command will write the option out for consumption by downstream tools.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

Some libraries contain multiple selection tables, such as for different numbers of metal layers). In such cases you can indicate which `wireload_selection` table should be used:

```
set_wire_load_selection_group [find / -wireload_selection "4_layer"]
```

Related Information

Sets this attribute:	wireload_selection
Related attributes:	force_wireload
	wireload_mode

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

sizeof_collection

`sizeof_collection` *`collection`*

Returns the total number of objects contained in the specified collection. If the collection contains different types of objects, `sizeof_collection` returns the sum of all the objects of different types in the collection.

This command is an efficient mechanism for determining the number of objects in a collection.

Options and Arguments

`collection`	Specifies a list whose length needs to be returned.

Example

- The following example shows a simple usage of the command:

```
legacy_genus:/> sizeof_collection [get_cells *]
4
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

sort_collection

```
sort_collection
     {clock|design|instance|library|libarc|
     libcell|libpin|net|pin|port}...
     property [property]...
     [-dictionary] [-descending] [-quiet]
```

Sorts the collection based on the specified properties. You can specify any property associated with the object type.

Options and Arguments

{clock|design|instance|library|libarc|libcell|libpin|net|pin|port}	Defines the collection to be sorted. The collection can only contain one type of objects.
`-descending`	Specifies to sort the objects in a descending order. By default, the objects are listed in ascending order.
`-dictionary`	Specifies to sort the objects by dictionary order.
property	Specifies the property to use to order the objects. If you specify more than one property, the command sorts the collection in the order by which you specified the properties. To get a list of properties, refer to the filter options of the corresponding get command. For example, for a list of properties of instances, refer to the `get_cells` command.
`-quiet`	Suppresses all the generated messages.

Examples

- The following command sorts the list of cells based on area and orders the cell based on increasing area.

```
legacy_genus:/> set list_cells [get_cells]
legacy_genus:/> sort_collection $list_cells area
```

```
/designs/m1/instances_comb/g127 /designs/m1/instances_comb/g132 /designs/m1/
instances_seq/o_m1_clk1_0_reg_0 /designs/m1/instances_seq/o_m1_clk1_0_reg_1
/designs/m1/instances_seq/o_m1_clk1_0_reg_2 /designs/m1/instances_hier/m2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

- The following command sorts the list of cells based on name and area.

```
legacy_genus:/> sort_collection $list_cells {name area}
/designs/m1/instances_comb/g127 /designs/m1/instances_comb/g132 /designs/m1/
instances_hier/m2 /designs/m1/instances_seq/o_m1_clk1_0_reg_0 /designs/m1/
instances_seq/o_m1_clk1_0_reg_1 /designs/m1/instances_seq/o_m1_clk1_0_reg_2
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

A

Applets

- Introduction on page 1654
- applet on page 1655
- applet avail on page 1656
- applet install on page 1658
- applet list on page 1659
- applet load on page 1660
- applet set_defaults on page 1662
- applet update on page 1663
- applet version on page 1664
- applet whatis on page 1665

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Introduction

Applets are **non-productized** scripts that can be used for a variety of purposes, such as report generation, template management, text histogram generation, test-case creation, and many others.

The applet infrastructure enables effective management of such infrastructure whether user-generated or Cadence-generated.

All applet commands generate a disclaimer banner, clearly indicating the limitation and conditions of their use.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet

```
applet {avail | install | list | load | set_defaults
     | update | version | whatis}
```

Manage applets.

Options and Arguments

`avail`	Lists all applets: those you have installed as well as those that are available on the server. The listing also includes the version information.
`install`	Installs the applet tree in the specified location.
`list`	Lists all applets already loaded.
`load`	Loads an applet and all of its dependencies.
`set_defaults`	Sets the `applet_mode`, `applet_server`, and `applet_search_path` attributes to their defaults.
`update`	Updates those applets for which a newer version exists on the applet server.
`version`	Returns the version information for the applet command.
`whatis`	Displays the description of a specific applet.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet avail

```
applet avail
     [-location applet_installation]
     [-outdated] [-local] [-detail]
```

Provides information for all the applets available in your local installation and those on the applet server, and includes version information allowing you to see version differences between installed applets and applets on the server.

Options and Arguments

`-detail`
: Reports the applets installed per installation: the list for your local installation, and the list available on the server.

`-location` *`applet_installation`*
: Specifies the location of the local applet installation from where applets can be loaded.

  Specify this option if you want to override the value of the `applet_search_path` root attribute.

  By default, the location specified through the `applet_search_path` attribute is used.

`-local`
: Only reports information for applets that exist in the local installation(s).

`-outdated`
: Only reports scripts for which a newer version is available on the server. New applets not present in the local installation are also listed.

Examples

- The following command lists all available applets.

```
legacy_genus:/> applet avail
============================================================================
...
============================================================================

Info: Collecting applet server information...
##############################################################################

Applets Local/Server information:
  Local Install (/home/me/.localApps/rc)
  Remote Server (<install>/tools.<platform>/lib/applets)

          Name          |  Installed  | Server Ver. | Summary Description
------------------------+-------------+-------------+--------------------------------------------
          compare_power |           6 |        1.15 | Generate HTML comparison report of power metrics..
           create_tcase |           5 |        1.42 | generate / load / save testcases
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

```
                 hermes |           3 |        1.14 | Hermes Design Assistant
                   lock |           5 |        1.12 | Guarantee atomic operations using a LOCKFILE...
          manual_assert |           3 |         1.3 | Apply default toggle rate to all synthesis ...

NOTE: To install an applet, use 'applet install -location <applets directory> <applet name>'
legacy_genus:/>
```

- The following command lists the outdated applets.

```
legacy_genus:/> applet avail -outdated
==========================================================================
...
==========================================================================

Info: Collecting applet server information...
##############################################################################

Applets Local/Server information:
  Local Install (/home/me/.localApps/rc)
  Remote Server (<install>/tools.<platform>/lib/applets)

          Name          |  Installed  | Server Ver. | Summary Description
------------------------+-------------+-------------+-----------------------------------------------------
          compare_power |           6 |         6.4 | Generate HTML comparison report of power metrics..
                   lock |         N/A |         1.3 | Guarantee atomic operations using a LOCKFILE, ...

NOTE: To install an applet, use 'applet install -location <applets directory> <applet name>'
legacy_genus:/>
```

- The following command lists the local applets.

```
legacy_genus:/> applet avail -local
==========================================================================
...
==========================================================================

Info: Collecting applet server information...
##############################################################################

Applets Local/Server information:
  Local Install (/home/me/.localApps/rc)
  Remote Server (<install>/tools.<platform>/lib/applets)

          Name          |  Installed  | Server Ver. | Summary Description
------------------------+-------------+-------------+-----------------------------------------------------
          compare_power |           6 |        1.13 | Generate HTML comparison report of power metrics..
           create_tcase |           5 |        1.42 | generate / load / save testcases
                 hermes |           3 |        1.14 | Hermes Design Assistant
          manual_assert |           3 |         1.3 | Apply default toggle rate to all synthesis ...

NOTE: To install an applet, use 'applet install -location <applets directory> <applet name>'
legacy_genus:/>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet install

```
applet install
     [-location applet_installation]
     [applet_list] [-force]
```

Installs one or more applets and their corresponding dependencies.

Important

An applet is not available for execution until it is loaded.

Options and Arguments

applet_list	Only installs the specified applets. By default, all applets on the server will be installed.
-force	Indicates to overwrite the installed applets. By default, the installed applets are not overwritten.
-location *applet_installation*	Specifies the location where to install the local applets. Specify this option if you want to override the value of the applet_search_path root attribute. By default, the location specified through the applet_search_path attribute is used.

Example

The following command installs the compare_gates applet in the specified location.

```
legacy_genus:/> applet install -location ~/rc_ex/applets compare_gates
Info: Collecting applet server information...
Installing applet 'compare_gates'...
Updating applet catalog information...
....
legacy_genus:/> applet install -location ~/rc_ex/applets compare_gates
Info: Collecting applet server information...
Installing applet 'compare_gates'...
Error: applet 'compare_gates' is already installed. Please use
        'update' command instead or '-force' switch
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet list

```
applet list [-tcl]
```

Displays all applets that have been loaded and that are available for use in the current session

Options and Arguments

`-tcl`	Returns the list of applets that were loaded as a Tcl list.

Example

The following example shows the applets that were loaded.

```
legacy_genus:/> applet load compare_power
...
legacy_genus:/> applet list
============================================================================
...
============================================================================

The following applets have been loaded:
        compare_power
        generate_report
        time_info


legacy_genus:/>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet load

```
applet load
     [-location applet_installation]
     [-no_init] [-server] [-version string]
     [applet_list]
```

Loads one or more applets and their corresponding dependencies.

Important

An applet is not available for execution until it is loaded.

Options and Arguments

applet_list	Only loads the specified applets. By default, all applets on the server will be loaded.
-location *applet_installation*	Specifies the location of the local applet installation from where applets can be loaded. Specify this option if you want to override the value of the `applet_search_path` root attribute. By default, the location specified through the `applet_search_path` attribute is used.
-no_init	Loads the applets without executing their init code. **Note:** The init code is optional for each applet.
-server	Loads the applet from the server instead of from the default installation. By default, the server and installation are the same (Genus installation tree). *Default:* Location specified in the `applet_search_path` root attribute in left to right priority order.
-version *string*	Specifies the version of the applet to load. This option can only be specified if only one applet is loaded.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Example

The following command loads the `compare_power` applet together with all its dependencies.

```
legacy_genus:/> applet load compare_power
============================================================================
...
============================================================================

Sourcing '/install_path/tools.lnx86/lib/applets/compare_power/compare_power.tcl'
(Tue Aug 23 15:52:34 -0700 2011)...
Sourcing '/install_path/tools.lnx86/lib/applets/generate_report/
generate_report.tcl' (Tue Aug 23 15:52:34 -0700 2011)...
Sourcing '/install_path/tools.lnx86/lib/applets/time_info/time_info.tcl' (Tue
Aug 23 15:52:34 -0700 2011)...
Loading applet 'compare_power'...
legacy_genus:/>
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet set_defaults

```
applet set_defaults
```

Sets the `applet_mode`, `applet_server`, and `applet_search_path` attributes to their defaults.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet update

```
applet update
     [-location applet_installation]
     applet_list
```

Compares the version and contents of the applets in the local installation with those on the server and updates any applets that have newer versions available on the server. Uninstalled applets need to be installed first.

Options and Arguments

`-location` *`applet_installation`*

Specifies the location of the local applet installation that must be updated.

Specify this option if you want to override the value of the `applet_search_path` root attribute.

By default, the location specified through the `applet_search_path` attribute is used.

`applet_list`

Only updates the specified applets.

If the specified applet was not yet installed, it will be installed in the specified location.

If you do not specify any applets, all applets on the local installation will be updated without installing any missing ones.

Examples

The following example shows a case where the *time_info* applet is newer on the server and hence updated on the local installation

```
legacy_genus:/> applet update
Info: Collecting applet server information...
Info: Updating applet installation /home/myname/.localApps/rc...
Updating applet 'time_info' (1.1 -> 1.2)...
Info: Connecting to server....
Info: Collecting applet server information...
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet version

`applet version [applet]`

Provides version information of the `applet` command or of the specified applet.

Options and Arguments

applet	Specifies the version of the specified applet. **Note:** This only applies to applets you have downloaded.

Example

- The following command shows the current version of the applet command.

```
legacy_genus:/> applet version
applet command Version: 4
```

- The following command returns the versions of the time_info applet.

```
legacy_genus:/> applet version time_info
applet 'time_info' Version: 1.52
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

applet whatis

`applet whatis [-detail] applet_list`

Provides a summary description of the capabilities of the specified applets.

Options and Arguments

applet_list	Lists the applets for which you want information.
-detail	Specifies to provide a detailed description of the capabilities of the specified applets.

Examples

- The following command requests summary information for the `time_info` and `lock` applets.

```
legacy_genus:/> applet whatis {time_info lock}
==============================================================================
disclaimer info...
==============================================================================

Info: Collecting applet server information...
Applet: time_info
        Location: install_path/lib/applets
        Version:  2
        Summary:  Create time stamp and keep track of runtime metrics

Applet: lock
        Location: install_path/lib/applets
        Version:  2
        Summary:  Guarantee atomic operations using a LOCKFILE, wrapper
functions
```

- The following command requests detailed information for the `time_info` applet.

```
legacy_genus:/> applet whatis time_info -detail
Info: Collecting applet server information...
Applet: time_info
        Location: install_path/lib/applets
        Version:  2
        Summary:  Create time stamp and keep track of runtime metrics

        Full Description:
                This script allows the user to create time stamps such that
                at any time the user can generate a report of the runtime
                allocation throughout a run.
```


© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

Index

Symbols

????????????? 1310
???????????????? 1313
????????????????????? 1311
????????????????????????????????????? 1311

A

abstract model
  logic, writing 404
  scan, reading 1113
  scan, writing 1162
adding
  design 1335
  directory to stack 73
  help for user command 1370
  user-defined command 1380
  user-defined message 1376
all_registers 1402
area, reporting 584
assigning, paths to cost group 485
ATPG file, writing out 864, 1148
Attribute
  defining new
    defining custom 1371
attributes
  retrieving value 101
  setting value 133

B

base cell, instantiating 1312
bitblasting
  one port 1298, 1320, 1323
bitblasting, all ports 1297, 1322
body segment, defining 966
boundary scan logic
  previewing changes 1028
busses
  renaming 1301

C

cells
  reporting cell count 584
  reporting technology library cells used 633
changing
  attribute value
    defined by RC 133
  directory in design hierarchy 53
  names
    of instances of an object type 1301
  path constraints 475
  UNIX working directory 109
channel masking types 859
clock
  removing 1485
clock gating
  insertion in mapped netlist 1230
clock inputs, reporting 567
clock waveform, defining 456
clock-gating logic
  editing 1221
  inserting and connecting observability logic 1231
  merging instances 1226
  removing 1236
  reporting 593
  test input, connecting 1225
color
  highlighting 232
  label 267
commands, alphabetical list of 31
compatible test clocks, declaring 1140
concatenating
  scan chains 897
configuring, pads for DFT 899
congestion map
  create snapshot for documentation 274
connect 1309
connecting
  pins, ports 1324
  scan chains 906
  test input of clock-gating logic 1225
constraints
  checking 711

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reading in 352
cost group
assigning paths to 485
defining 461
create_design 1310
creating
binding for Chipware component 299
Chipware component 301
delay constraint for specific path 479
design 1335
design representation in design hierarchy 505
HDL library 305
hierarchy in design 1332, 1345
implementation for Chipware component 303
instance of 1312, 1336
library domains (MSV) 1282
module 1317
parameter for Chipware component 309
pin 311
port bus 1338
subdesign 1326
subport bus 1341
synthetic operator 306
timing constraints for instance 462
user-defined message 1376
critical path, reporting timing slack for 709
current directory
in design hierarchy 74
in UNIX 120

D

define_dft pmbist_port 962
defining
boundary scan segment 921
clock 456
configuration mode 925
cost group 461
DFT test clock 944, 990
input and output delays 464
multi-cycle path 471
paths for timing constraints 494
scan clock for LSSD 972, 975
shift-enable signal for DFT 978
test-mode signal for DFT 995
user_defined scan chain 966
user-defined abstract segment 915
design
area 584
creating
new design 1335
instantiating 1312, 1336
removing 1349
renaming 1301
uniquifying 1343, 1359
design hierarchy
adding hierarchy level 1332, 1345
changing directories in 53, 1416, 1418
finding object type 60
listing information 67
removing hierarchy level 1342
removing objects 1320, 1328, 1355
renaming objects 1349
returning current position 74
design rule constraints
reporting violations on 608
DFT clock domain
associating with scan chain 967
specifying compatible test clocks 1140
DFT MBIST clocks
defining 944
DFT rule violations
checking flip-flops for 870
fixing 1010
DFT rules, list of rules checked 870
DFT test clocks
declaring as compatible 1140
defining 990
directories
changing in
design hierarchy 53
UNIX 109
listing contents in
design hierarchy 67
UNIX directory 116
returning current position in
design hierarchy 74
UNIX 120
directory stack
adding new directory 73
displaying directory list 56
tracing previous directory 71
disconnect 1319
disconnecting pins, ports 1329
displaying
current position in design hierarchy 74
directory stack 56
UNIX directory, contents 116

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

E

executing
  script 106
  UNIX shell command 136, 137
exiting, from Genus 100

F

fanin, reporting 572
fanout, reporting 576
filtering, objects 57
find, object type 60
finding
  corresponding input or output 66
  object base name 52
  object directory name 55
  object type 60
  type of object 79
fixing, DFT rule violations 1010
flattening, instances 1357
flip-flops
  checking for DFT rule violations 870
  reporting scannability 620
Floorplan (DEF)
  Reading 814
  Writing 837

G

Genus
  exiting 100
  quit 125
  resuming process 144
  suspending process 144
get
  library domains 1362
GUI
  update 224

H

HDL files, reading 336
HDL Viewer
  remove data 226
head segment, defining 968
help
  adding for new command 1370
  providing for commands 86, 105
highlight
  clear 230, 250
  color 232, 263

I

input delay, defining 464
inserting
  clock-gating logic in mapped netlist 1230
  shadow logic 1077
  test point 1083
  user-defined test point 1089
  wrapper cell 1094
instances
  grouping 1332, 1345
  removing from design 1349
  renaming 1301
  ungrouping 1342
isolation rule
  removing 1349

L

LBIST logic
  previewing changes 1057
leakage power, reporting 669
length
  abstract segment 916
  maximum chain length 969
level-shifter groups
  removing 1349
library cell, instantiating 1312, 1336
library domains
  creating 1282
licenses
  checked out, listing 114
  checking out 112
listing
  object information 67
load unit, specifying in SDC file 1592
lockup element type, chain-specific 970
logic abstract model, definition 404
logic abstract, writing 404
loop breakers
  removing 1351
  reporting 590

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

M

mapped netlist
writing out 404
mapping
non-scan flops with scan-equivalent 1123
memory resources, reporting 642
messages
creating customized 1376
reporting 644
sending 1378
suppressing 142
mode, specifying for power, timing analysis, and optimization 453
module
creating by copying existing module 1317
MSV design
library domains, creating 1282
multi-cycle path
defining 1621
multi-cycle path, defining 471

N

names
adding prefix 1303
limiting length 1303
mapping characters 1303
net power, reporting 669
nets
renaming 1301

O

object
finding base name 52
finding directory name 55
object path
return a selection list for instance, net, pin, and port objects 222
object type, finding 60
object types
listing all attributes for a type 101, 133
objects
connecting 1324
disconnecting 1329
filtering on attribute value 57
instantiating 1312, 1336
listing information for an object 67
removing from design hierarchy 1320, 1328, 1355
renaming 1349
returning type of an object 65, 79
selecting in design hierarchy 57
observability logic for clock-gating logic
inserting 1231
OPCG logic
previewing changes 1063
OPCG segments
previewing connections 904
optimizing
for area or timing 519
output delay, defining 464
output, redirecting 126

P

parameterized module, elaborating 505
path
return path for a selected object 222
path constraints, modifying 475
paths
assigning to cost group 485
creating for timing exception 494
unconstraining 482
pins
connecting manually 1325
disconnecting 1329
port buses
creating new object 1338
removing from design 1349
renaming 1301
ports
bitblasting 1297, 1322
connecting manually 1325
create 1313
disconnect 1319
disconnecting 1329
renaming 1301
power
reporting 669
sorting report 672
power domain
removing 1349
primitive cell, creating 1314, 1339
probability values of nets

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

reading or updating 1247
writing to SAIF file 1263
writing to TCF file 1265

R

read
qrc
extraction 350
reading
HDL files 336
SAIF file 1242
SDC constraints 352
TCF file 1247
remove
data 226
removing
clock 1485
clock-gating logic 1236
directory from stack 71
hierarchy from design 1342
object from design hierarchy 1328, 1355
renaming
instances of object type 1301
object 1349
reporting
area of design 584
cell types, used 633
clock information of design 600
clock-gating information 593
design rule violations 608
DFT registers 620
DFT setup information 624
DFT violations 629
inferred datapath operators 603
instance information 639
memory resources 642
messages in current session 644
net information 661
port information 668
scan chains 610
timing information 709
reset_max_time_borrow 1506
retrieving
attribute value
defined by RC 101
clock ports 567
fanin information 572
fanout information 576
object information 67
UNIX working directory 120
RTL Compiler
resuming process 144
RTL power analysis, enabling 671

S

SAIF file
reading 1242
writing 1261
scan abstract model 731
scan abstract model, creating 1162
scan abstract model, reading 1113
scan chain
tool-created
test clock domain associated with name 929, 931, 963, 981, 1019
scan-chain configuration
previewing 909
reporting 610
scan chains
connecting 906
defining 966
mixing edges of same clock on 1140
removing 1349
reporting 610
scan data input
creating 967
specifying for chain 969
specifying for scan segment 917, 964
using existing port 969
scan data output
creating 967
specifying for chain 969
specifying for scan segment 917, 964
using existing port 893, 970, 997, 1005, 1039
scan segment
defining 915, 929, 931, 963, 981, 1019
removing 1349
shift enable
confirming if connected 916, 964
using in scan chain
as body segment 966
as head segment 968
as tail segment 970
scanDEF file, writing out 1216
scripts
executing 106

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.

writing out 419
SDC constraints
reading in 352
unsupported constructs 352
writing out 422
shadow logic
inserting 1077
previewing changes 1079
shift-enable port
for scan segment 918
confirming if connected 916, 964
for shift-enable signal 954, 980
shift-enable signal
defining 978, 995
specifying default 978, 1001
specifying for chain 970
slack, reporting at timing endpoints 710
SPEF
Reading 822
Writing 841
subdesign, instantiating 1312
subdesigns
creating by copying existing subdesign 1326
instantiating 1336
renaming 1301
uniquifying 1343, 1359
subport buses
creating new object 1341
removing from design 1349
renaming 1301
subports
connecting manually 1325
disconnect 1319
disconnecting 1329
switching power, reporting 669

T

tail segment, defining 970
TCF file
reading 1247
writing 1265
test mode signal, defining 995
test point
inserting 1083
possible types 1085
time unit
specifying in SDC file 1641
timing constraints
deriving for instance 462
reporting violations on 709
writing out 419
timing exception, created by
modifying path constraints 475
overriding default clock edge relationship 471
specifying timing constraints 479
unconstraining paths 482
timing exceptions
creating 1621
timing units 709
toggle counts of nets
reading or updating 1247
writing to SAIF file 1263
writing to TCF file 1265

U

uniquifying, design or subdesign 1343, 1359
UNIX shell command
executing in Genus 136, 137
user-defined test point, inserting 1089
utilization map
create snapshot for documentation 275

W

wireload model, reporting 584
worst paths, reporting timing for 713
wrapper cell, inserting 1094
writing
compression model 1154
DFT abstract 1162
logic abstract 404
netlist file 404
SAIF file 1261, 1263
scanDEF file 1216
SDC constraints 422
TCF file 1265
timing and design rule constraints 419

© 2015-2019 Cadence Design Systems, Inc. All Rights Reserved.