Contents

Alphabetical List of Commands ................................. 21

Preface .............................................................. 27
About This Manual ................................................. 28
Additional References ............................................. 28
How to Use the Documentation Set .............................. 29
Reporting Problems or Errors in Manuals ...................... 30
Customer Support ................................................ 31
  Cadence Online Support ...................................... 31
  Other Support Offerings ...................................... 31
Messages .......................................................... 32
Man Pages ........................................................ 33
Command-Line Help .............................................. 34
  Getting the Syntax for a Command ......................... 34
  Getting the Syntax for an Attribute ...................... 34
  Searching for Attributes ................................... 35
  Searching For Commands When You Are Unsure of the Name 35
Documentation Conventions ..................................... 36
  Text Command Syntax ..................................... 36

1 Navigation ......................................................... 37
  basename ....................................................... 38
  cd ............................................................... 39
  dirname ........................................................ 41
  dirs ............................................................. 42
  filter ........................................................... 43
  find ............................................................ 46
  inout_mate .................................................... 51
  l ................................................................. 52
Command Reference for Encounter RTL Compiler

ls ................................................................. 53
popd ............................................................ 58
pushd ........................................................... 59
pwd ............................................................. 60
vdir_isearch .................................................. 61
vname .......................................................... 63
what_is ......................................................... 64
what_is_list ................................................... 65

2
General ......................................................... 67
?

69

all_inputs ..................................................... 71
all_outputs .................................................... 72
apropos ......................................................... 73
attribute_exists ............................................. 74
clear ............................................................ 75
clear_redline_terminal .................................... 76
date ............................................................. 77
enable_transparent_latches ............................... 78
exec_embedded_script ...................................... 79
exit .............................................................. 81
get_attribute ................................................ 82
get_liberty_attribute ...................................... 84
get_read_files .............................................. 85
help .............................................................. 86
include ........................................................ 87
cd ................................................................. 88
license ........................................................ 89
license_checkin .............................................. 90
license_checkout ............................................ 91
license_feature ............................................. 93
license_list .................................................. 94
license_version ............................................. 95
3

**GUI Text**

General GUI Text Commands

- gui_balloon_info ........................................ 125
- gui_hide .................................................. 125
- gui_info .................................................. 125
- gui_legend ............................................... 126
- gui_raise ................................................ 127
- gui_reset ............................................... 127
- gui_resume ............................................. 128
- gui_selection ......................................... 128
- gui_show ................................................. 128
- gui_status .............................................. 129
gui_suspend .......................................................... 129
gui_update .......................................................... 129

HDL Viewer GUI Text Commands .................................... 130
gui_hv_clear ....................................................... 131
gui_hv_get_file .................................................. 131
gui_hv_load_file ................................................. 131
gui_hv_set_indicators .......................................... 132

Schematic Viewer GUI Text Commands .............................. 133
gui_sv_clear ...................................................... 134
gui_sv_cone ....................................................... 134
gui_sv_get_instance ............................................. 134
gui_sv_grey ......................................................... 134
gui_sv_highlight ............................................... 135
gui_sv_load ......................................................... 136
gui_sv_snapshot .................................................. 137
gui_sv_toolbar_button ......................................... 137

Physical Viewer GUI Text Commands ............................... 138
gui_pv_airline_add ............................................... 140
gui_pv_airline_add_custom ..................................... 142
gui_pv_airline_delete .......................................... 143
gui_pv_airline_display ......................................... 144
gui_pv_airline_raw_add ....................................... 145
gui_pv_airline_raw_add_custom ............................... 146
gui_pv_align_instance_to_boundary .......................... 147
gui_pv_align_instances ......................................... 148
gui_pv_clear ...................................................... 148
gui_pv_connectivity_airlines .................................. 149
gui_pv_deselect .................................................. 150
gui_pv_display_collection ..................................... 151
gui_pv_draw_box ............................................... 152
gui_pv_draw_circle ............................................. 153
gui_pv_draw_line ................................................ 154
gui_pv_draw_triangle ......................................... 155
gui_pv_get_design ............................................... 156
gui_pv_grey ........................................................ 156
gui_pv_highlight ............................................... 157
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gui pv highlight hier instances</td>
<td>159</td>
</tr>
<tr>
<td>gui pv highlight update</td>
<td>160</td>
</tr>
<tr>
<td>gui pv label</td>
<td>161</td>
</tr>
<tr>
<td>gui pv lineup instances</td>
<td>162</td>
</tr>
<tr>
<td>gui pv new viewer</td>
<td>162</td>
</tr>
<tr>
<td>gui pv preferences</td>
<td>163</td>
</tr>
<tr>
<td>gui pv redraw</td>
<td>164</td>
</tr>
<tr>
<td>gui pv select</td>
<td>164</td>
</tr>
<tr>
<td>gui pv selection</td>
<td>165</td>
</tr>
<tr>
<td>gui pv snapshot</td>
<td>166</td>
</tr>
<tr>
<td>gui pv steiner tree</td>
<td>168</td>
</tr>
<tr>
<td>gui pv toolbar button</td>
<td>169</td>
</tr>
<tr>
<td>gui pv update</td>
<td>170</td>
</tr>
<tr>
<td>gui pv zoom box</td>
<td>171</td>
</tr>
<tr>
<td>gui pv zoom fit</td>
<td>171</td>
</tr>
<tr>
<td>gui pv zoom in</td>
<td>172</td>
</tr>
<tr>
<td>gui pv zoom out</td>
<td>172</td>
</tr>
<tr>
<td>gui pv zoom to</td>
<td>173</td>
</tr>
</tbody>
</table>

4

**Chipware Developer**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwd</td>
<td>175</td>
</tr>
<tr>
<td>cwd check</td>
<td>176</td>
</tr>
<tr>
<td>cwd create_check</td>
<td>177</td>
</tr>
<tr>
<td>cwd create_check</td>
<td>181</td>
</tr>
<tr>
<td>cwd report_check</td>
<td>183</td>
</tr>
<tr>
<td>hdl create</td>
<td>185</td>
</tr>
<tr>
<td>hdl create binding</td>
<td>186</td>
</tr>
<tr>
<td>hdl create component</td>
<td>188</td>
</tr>
<tr>
<td>hdl create implementation</td>
<td>190</td>
</tr>
<tr>
<td>hdl create library</td>
<td>192</td>
</tr>
<tr>
<td>hdl create operator</td>
<td>193</td>
</tr>
<tr>
<td>hdl create package</td>
<td>194</td>
</tr>
<tr>
<td>hdl create parameter</td>
<td>196</td>
</tr>
<tr>
<td>hdl create pin</td>
<td>198</td>
</tr>
</tbody>
</table>
5
Input and Output ................................................................. 201

decrypt ................................................................. 204
encrypt ............................................................... 205
export_critical_endpoints ............................................ 208
read_db .............................................................. 210
read_def ............................................................ 211
read_dfm ............................................................ 212
read_dft_abstract_model ............................................ 214
read_encounter ..................................................... 215
read_hdl ............................................................. 216
read_io_speclist ................................................... 220
read_memory_view .................................................. 221
read_netlist ........................................................ 222
read_pmbist_interface_files ..................................... 224
read_power_intent .................................................. 225
read_saif .......................................................... 226
read_sdc ........................................................... 227
read_spef .......................................................... 229
read_tcf ........................................................... 230
read_vcd ........................................................... 231
restore_design ....................................................... 232
split_db ............................................................. 234
write_atpg .......................................................... 235
write_bsdld ......................................................... 236
write_compression_macro ......................................... 237
write_db ............................................................ 238
write_def .......................................................... 240
write_design ......................................................... 241
write_dft_abstract_model ......................................... 243
write_dft_rtl_model ................................................ 244
write_do_ccd ....................................................... 245
write_do_ccd_cdc .................................................. 246
write_do_ccd_compare_sdc ....................................... 247
write_do_ccd_generate ............................................ 249
write_do_ccd_propagate ........................................ 251
write_do_ccd_validate ........................................ 253
write_do_clp ....................................................... 254
write_do_lec ....................................................... 257
write_do_verify_cdc ........................................ 261
write_encounter ................................................... 263
write_et_atpg ..................................................... 266
write_et_bsv ....................................................... 267
write_et_dfa ....................................................... 268
write_et_lbist .................................................... 269
write_et_mbist ................................................... 270
write_et_no_tp_file ............................................ 271
write_et_rrfa ...................................................... 272
write_ets .......................................................... 273
write_ett ........................................................... 274
write_forward_saif ............................................. 275
write_hdl ........................................................... 276
write_io_speclist .................................................. 280
write_ldb ........................................................... 281
write_logic_bist_macro ......................................... 282
write_mbist_testbench .......................................... 283
write_pmbist_interface_files ................................. 284
write_pmbist_testbench ........................................ 285
write_power_intent ............................................... 286
write_saif .......................................................... 287
write_scandef ..................................................... 288
write_script ......................................................... 289
write_sdc .......................................................... 292
write_sdf ........................................................... 296
write_set_load ..................................................... 301
write_specf ......................................................... 302
write_sv_wrapper .................................................. 303
write_tcf ........................................................... 306
write_template ..................................................... 307
6  Constraints ................................................................. 311
   clock_uncertainty .................................................. 312
   create_mode ........................................................ 315
   define_clock ....................................................... 318
   define_cost_group ............................................... 323
   derive_environment .............................................. 324
   external_delay .................................................... 326
   generate_constraints ............................................ 330
   multi_cycle ....................................................... 332
   path_adjust ........................................................ 336
   path_delay ........................................................ 340
   path_disable ...................................................... 343
   path_group ........................................................ 346
   propagate_constraints ........................................... 349
   specify_paths .................................................... 351
   validate_constraints ............................................ 357

7  Elaboration and Synthesis .......................................... 359
   elaborate .......................................................... 360
   get_remove_assign_options ..................................... 363
   merge_to_multibit_cells ........................................ 364
   remap_to_dedicated_clock_library .............................. 365
   remove_assigns_without_optimization .......................... 366
   remove_clock_reconvergence ................................... 369
   remove_inserted_sync_enable_logic ............................ 370
   retime ............................................................. 371
   set_remove_assign_options ..................................... 373
   synthesize ....................................................... 377

8  Hierarchical Flow ....................................................... 383
   assemble_design .................................................. 384
generate ilm ................................................................. 385
read ilm ................................................................. 387
read ilm from files .................................................... 388

9
Analysis and Report .................................................. 389
all_connected ............................................................ 392
all des ................................................................. 393
all des inps ............................................................. 394
all des insts ............................................................ 395
all des outs ............................................................. 396
all des seqs ............................................................. 397
all lib ................................................................. 399
all lib bufs ............................................................. 400
all lib ties ............................................................. 401
analyze_library_corners ............................................ 402
check_design .......................................................... 404
clock_ports ............................................................ 409
compare_sdc ........................................................... 410
create_timing_bin ...................................................... 411
fanin ................................................................. 413
fanout ................................................................. 417
report ................................................................. 420
report area ............................................................. 425
report boundary_opto ................................................. 427
report_case_analysis .................................................. 429
report_cdn_loop_breaker ............................................. 430
report_cell_delay_calculation ...................................... 431
report_clock_gating ................................................... 432
report_clocks .......................................................... 438
report_congestion ...................................................... 440
report_datapath ......................................................... 441
report_design_rules .................................................... 446
report_dft_chains ....................................................... 448
report_dft_clock_domain_info ...................................... 453
report dft_core_wrapper .................................................. 454
report dft_registers ...................................................... 458
report dft_setup ......................................................... 462
report dft_violations .................................................... 466
report disabled_transparent_latches .................................. 469
report gates .............................................................. 470
report hierarchy .......................................................... 474
report instance ............................................................ 476
report low_power_cells .................................................. 479
report low_power_intent .................................................. 482
report memory ............................................................. 486
report memory_cells ...................................................... 487
report messages ........................................................... 488
report mode ............................................................... 490
report multibit_inferencing .............................................. 491
report net_cap_calculation .............................................. 496
report net_delay_calculation ............................................ 497
report net_res_calculation .............................................. 498
report nets ................................................................. 499
report opcg_equivalents .................................................. 502
report ple ................................................................. 503
report port ................................................................. 505
report power ............................................................... 507
report power_domain ...................................................... 519
report proto ............................................................... 521
report qor ................................................................. 523
report scan_compressibility ............................................. 529
report sequential .......................................................... 531
report slew_calculation .................................................. 534
report summary ............................................................ 535
report test_power .......................................................... 537
report timing ............................................................... 541
report units ............................................................... 552
report utilization .......................................................... 553
report yield ............................................................... 554
statistics ................................................................. 555
statistics add_metric ................................................................. 557
statistics log ................................................................. 558
statistics read ............................................................... 560
statistics remove_metric ....................................................... 561
statistics report ............................................................... 562
statistics reset ................................................................. 565
statistics run_stage_ids ......................................................... 566
statistics write ................................................................. 567
timestat .............................................................. 568
validate_timing ................................................................. 569

10
Physical ................................................................. 571
check_floorplan ................................................................. 573
check_placement ................................................................. 577
create_group ................................................................. 579
create_placement_blockage .................................................... 580
create_placement_halo_blockage ............................................. 581
create_region ................................................................. 582
create_row ................................................................. 584
create_routing_blockage ....................................................... 585
create_routing_halo_blockage ................................................. 587
create_track ................................................................. 588
def_move ................................................................. 589
duplicate_register ........................................................... 590
generate_ple_model ............................................................ 592
generate_reports .............................................................. 594
modify_power_domain_attr .................................................... 596
move_blockage ................................................................. 598
move_instance ................................................................. 599
move_port ................................................................. 600
move_region ................................................................. 601
read_def ................................................................. 602
read_encounter ............................................................... 606
read_sdp_file ................................................................. 607
Command Reference for Encounter RTL Compiler

11

Design for Test ................................................................. 627

add_opcg_hold_mux ............................................................ 632
analyze_scan_compressibility ................................................. 633
analyze_testability ............................................................. 642
check_atpg_rules ............................................................... 645
check_dft_pad_configuration ............................................... 647
check_dft_rules ............................................................... 648
check_mbist_rules ............................................................. 654
compress_block_level_chains ................................................. 657
compress_scan_chains ........................................................ 660
concat_scan_chains ............................................................ 675
configure_pad_dft ............................................................. 677
connect_compression_clocks ................................................... 678
connect_opcg_segments ....................................................... 679
connect_scan_chains .......................................................... 681
define_dft ................................................................. 686
define_dft abstract_segment .................................................. 689
define_dft boundary_scan_segment ........................................... 695
define_dft dft_configuration_mode ........................................ 699
define_dft domain_macro_parameters ...................................... 702
define_dft fixed_segment ..................................................... 704
define_dft floating_segment ............................................... 706
define_dft jtag_instruction .................................................. 708
define_dft jtag_instruction_register ..................................... 712
define_dft jtag_macro .......................................................... 714
define_dft mbist_clock ...................................................... 719
define_dft mbist_direct_access ............................................. 722
define_dft opcg_domain ...................................................... 725
define_dft opcg_mode ........................................................ 728
define_dft opcg_trigger ..................................................... 730
define_dft osc_source .......................................................... 732
define_dft pmbist_direct_access ............................................ 734
define_dft preserved_segment .............................................. 736
define_dft scan_chain ......................................................... 739
define_dft scan_clock_a ...................................................... 745
define_dft scan_clock_b ...................................................... 748
define_dft shift_enable ....................................................... 751
define_dft shift_register_segment ......................................... 754
define_dft tap_port ............................................................. 756
define_dft test_bus_port ....................................................... 758
define_dft test_clock .......................................................... 762
define_dft test_mode ........................................................... 766
dft_trace_back ................................................................. 770
fix_dft_violations ............................................................... 772
fix_scan_path_inversions ..................................................... 776
identify_domain_crossing_pins_for_cgif_and_scan_abstracts .......... 777
identify_multibit_cell_abstract_scan_segments .......................... 778
identify_shift_register_scan_segments .................................... 780
identify_test_mode_registers ............................................... 782
insert_dft ....................................................................... 785
insert_dft_boundary_scan ................................................... 788
insert_dft_compression_logic .............................................. 793
insert_dft dfa_test_points ........................................... 804
insert_dft jtag_macro .................................................. 808
insert_dft lockup_element ........................................... 812
insert_dft logic_bist .................................................. 813
insert_dft mbist ......................................................... 819
insert_dft opcg .......................................................... 825
insert_dft pmbist ......................................................... 827
insert_dft ptam ............................................................ 833
insert_dft rfa_test_points ............................................ 836
insert_dft scan_power_gating ....................................... 843
insert_dft shadow_logic ............................................... 846
insert_dft shift_register_test_points .............................. 851
insert_dft test_point .................................................... 852
insert_dft user_test_point ............................................. 858
insert_dft wir_signal_bits ........................................... 860
insert_dft wrapper_cell ............................................... 861
insert_dft wrapper_instruction_register ............................ 867
insert_dft wrapper_mode_decode_block ............................. 869
insert_test_compression ............................................... 871
map_mbist_cgc_to_cgic ............................................... 874
read_dft_abstract_model ............................................. 875
read_io_speclist ....................................................... 878
read_memory_view ...................................................... 879
read_pmbist_interface_files ....................................... 881
replace_opcg_scan .................................................... 882
replace_scan ............................................................. 884
report_dft_chains ..................................................... 885
report_dft_clock_domain_info ..................................... 886
report_dft_core_wrapper ........................................... 887
report_dft_registers ................................................ 888
report_dft_setup ....................................................... 889
report_dftViolations .................................................. 890
report_opcg_equivalents ............................................ 891
report_scan_compressibility ....................................... 892
reset_opcg_equivalent ............................................... 894
<table>
<thead>
<tr>
<th>Command Reference for Encounter RTL Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset scan equivalent ................................ 895</td>
</tr>
<tr>
<td>set compatible test clocks ................................. 896</td>
</tr>
<tr>
<td>set opcg equivalent ........................................ 898</td>
</tr>
<tr>
<td>set scan equivalent ......................................... 900</td>
</tr>
<tr>
<td>update scan chains ........................................... 902</td>
</tr>
<tr>
<td>write atpg .................................................. 904</td>
</tr>
<tr>
<td>write bsdl .................................................. 907</td>
</tr>
<tr>
<td>write compression macro .................................... 910</td>
</tr>
<tr>
<td>write dft abstract model ................................... 918</td>
</tr>
<tr>
<td>write dft rtl model ......................................... 921</td>
</tr>
<tr>
<td>write et atpg ................................................ 922</td>
</tr>
<tr>
<td>write et bsv ................................................ 930</td>
</tr>
<tr>
<td>write et dfa ................................................ 934</td>
</tr>
<tr>
<td>write et lbist .............................................. 938</td>
</tr>
<tr>
<td>write et mbist .............................................. 942</td>
</tr>
<tr>
<td>write et no tp file ....................................... 947</td>
</tr>
<tr>
<td>write et rrfa ............................................... 948</td>
</tr>
<tr>
<td>write io specialist ..................................... 952</td>
</tr>
<tr>
<td>write logic bist macro ..................................... 954</td>
</tr>
<tr>
<td>write mbist testbench ..................................... 958</td>
</tr>
<tr>
<td>write pmbist interface files .............................. 962</td>
</tr>
<tr>
<td>write pmbist testbench ..................................... 964</td>
</tr>
<tr>
<td>write scandef ............................................... 968</td>
</tr>
</tbody>
</table>

12

Low Power Synthesis ............................................ 971

<table>
<thead>
<tr>
<th>Command Reference for Encounter RTL Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>build rtl power models ................................. 973</td>
</tr>
<tr>
<td>clock gating ............................................... 975</td>
</tr>
<tr>
<td>clock gating connect_test ............................... 977</td>
</tr>
<tr>
<td>clock gating declone ...................................... 978</td>
</tr>
<tr>
<td>clock gating import ....................................... 979</td>
</tr>
<tr>
<td>clock gating insert in netlist ........................... 981</td>
</tr>
<tr>
<td>clock gating insert obs .................................. 982</td>
</tr>
<tr>
<td>clock gating join .......................................... 984</td>
</tr>
<tr>
<td>clock gating remove ....................................... 986</td>
</tr>
</tbody>
</table>
clock_gating share .................................................. 988
clock_gating split .................................................. 990
read_saif .......................................................... 992
read_tcf .......................................................... 997
read_vcd .......................................................... 1002
report clock_gating .............................................. 1006
report power ...................................................... 1007
state_retention .................................................... 1008
state_retention connect_power_gating_pins .................. 1009
state_retention swap .............................................. 1010
write_forward_saif .............................................. 1011
write_saif ........................................................ 1013
write_tcf ........................................................ 1015

13
Advanced Low Power Synthesis .................................. 1017
apply_power_intent ................................................ 1018
check_cpf ........................................................ 1021
check_library ..................................................... 1024
commit_power_intent .............................................. 1030
create_library_domain .......................................... 1031
read_power_intent ................................................ 1032
report_low_power_cells ........................................ 1034
report_low_power_intent ........................................ 1035
verify_power_structure ......................................... 1036
write_power_intent ............................................... 1038

14
Design Manipulation ................................................ 1041
change_link ....................................................... 1043
change_names ...................................................... 1045
clock_gating ....................................................... 1052
delete_unloaded_undriven ....................................... 1053
edit_netlist ....................................................... 1054
edit_netlist bitblast_all_ports ................................ 1056
edit_netlist bitblast_port ................................................. 1057
edit_netlist connect ..................................................... 1058
edit_netlist dedicate_subdesign ....................................... 1060
edit_netlist delete ...................................................... 1061
edit_netlist disconnect ............................................... 1062
edit_netlist group ....................................................... 1064
edit_netlist hier_connect ............................................. 1066
edit_netlist new_design ................................................ 1067
edit_netlist new_instance ............................................. 1068
edit_netlist new_port_bus ............................................. 1070
edit_netlist new_primitive ............................................ 1071
edit_netlist new_subport_bus ....................................... 1073
edit_netlist ungroup ................................................... 1074
edit_netlist uniquify ................................................... 1075
group ........................................................................ 1076
insert_tiehilo_cells .................................................... 1077
mv ........................................................................... 1080
remove_cdn_loop_breaker ........................................... 1082
reset_design .............................................................. 1084
rm ........................................................................... 1085
ungroup .................................................................... 1086
uniquify ................................................................... 1088

15
Customization ............................................................. 1089
add_command_help ....................................................... 1090
define_attribute .......................................................... 1091
mesg_make ................................................................. 1096
mesg_send ................................................................. 1098
parse_options ............................................................. 1099

A
Applets ......................................................................... 1103
Introduction .................................................................. 1104
applet ........................................................................ 1105
Command Reference for Encounter RTL Compiler

applet avail ................................................................. 1106
applet install ............................................................. 1108
applet list ................................................................. 1109
applet load ................................................................. 1110
applet set_defaults ....................................................... 1112
applet update ............................................................ 1113
applet version ............................................................. 1114
applet whatis .............................................................. 1115

Index .................................................................................. 1
Alphabetical List of Commands

Symbols

? 69

A

add_command_help 1090
add_opcg_hold_mux 632
alias 70
all des inps 394
all des insts 395
all des outs 396
all des seqs 397
all lib 399
all lib bufs 400
all lib ties 401
all_connected 392
all_inputs 71
all_outputs 72
analyze_library_corners 402
analyze_scan_compressibility 633
analyze_testability 642
applet 1105
applet avail 1106
applet install 1108
applet list 1109
applet load 1110
applet_set_defaults 1112
applet update 1113
applet version 1114
applet whatis 1115
apply_power_intent 1018
apropos 73
assemble_design 384
attribute_exists 74

B

basename 38
build_rtl_power_models 973

C

cd 39
change_link 1043
change_names 1045
check_atpg_rules 645
check_cpf 1021
check_design 404
check_dft_pad_configuration 647
check_dft_rules 648
check_floorplan 573
check_library 1024
check_mbist_rules 654
check_placement 577
clear 75
clear_redline_terminal 76
clock_gating 975
clock_gating_connect_test 977
clock_gating_declone 978
clock_gating_import 979
clock_gating_insert_in_netlist 981
clock_gating_insert_obs 982
clock_gating_join 984
clock_gating_remove 986
clock_gating_share 988
clock_gating_split 990
clock_ports 409
clock_uncertainty 312
commit_power_intent 1030
compare_sdc 410
compress_block_level_chains 657
compress_scan_chains 660
concat_scan_chains 675
configure_pad_dft 677
connect_compression_clocks 678
connect_opcg_segments 679
connect_scan_chains 681
create_group 579
create_library_domain 1031
create_mode 315
create_placement_blockage 580
create_placement_halo_blockage 581
create_region 582
create_routing_blockage 585
create_row 584
create_timing_bin 411
gui_hv_get_file 131
gui_hv_load_file 131
gui_hv_set_indicators 132
gui_info 125
gui_legend 126
gui_pv_airline_add 140
gui_pv_airline_add_custom 142
gui_pv_airline_delete 143
gui_pv_airline_display 144
gui_pv_airline_raw_add 145
gui_pv_airline_raw_add_custom 146
gui_pv_align_instance_to_boundary 147
gui_pv_align_instances 148
gui_pv_clear 148
gui_pv_connectivity_airlines 149
gui_pv_deselect 150
gui_pv_display_collection 151
gui_pv_draw_box 152
gui_pv_draw_circle 153, 155
gui_pv_draw_line 154
gui_pv_get_design 156
gui_pv_grey 156
gui_pv_highlight 157
gui_pv_highlight_hier_instance 159
gui_pv_highlight_update 160
gui_pv_label 161
gui_pv_lineup_instances 162
gui_pv_new_viewer 162
gui_pv_preferences 163
gui_pv_redraw 164
gui_pv_select 164
gui_pv_selection 165
gui_pv_snapshot 166
gui_pv_steiner_tree 168
gui_pv_toolbar_button 169
gui_pv_update 170
gui_pv_zoom_box 171
gui_pv_zoom_fit 171
gui_pv_zoom_in 172
gui_pv_zoom_out 172
gui_pv_zoom_to 173
gui_raise 127
gui_reset 127
gui_resume 128
gui_selection 128
gui_show 128
gui_status 129
gui_suspend 129
gui_sv_clear 134
gui_sv_cone 134
gui_sv_get_instance 134
gui_sv_grey 134
gui_sv_highlight 135
gui_sv_load 136
gui_sv_snapshot 137
gui_sv_toolbar_button 137
gui_update 129

H

hd1_create 185
hd1_create binding 186
hd1_create component 188
hd1_create implementation 190
hd1_create library 192
hd1_create operator 193
hd1_create package 194
hd1_create parameter 196
hd1_create pin 198
help 86

I

identify_domain_crossing_pins_for_cgic_and
identify_domain_crossing_pins_for_cgic_and_scan_abstracts 777
identify_domain_crossing_pins_for_cgic_and_scan_abstracts 777
identify_multibit_cell_abstract_scan_segments 778
identify_shift_register_scan_segments 780
identify_test_mode_registers 782
include 87
inout_mate 51
insert_dft 785
insert_dft boundary_scan 788
insert_dft compression_logic 793
insert_dft jtag_macro 808
insert_dft lockup_element 812
insert_dft logic_bist 813
insert_dft mbist 819
insert_dft opcg 825
insert_dft pmbist 827
insert_dft ptam 833
insert_dft rfa_test_points 836
insert_dft scan_power_gating 843
insert_dft shadow_logic 846
insert_dft shift_register_test_points 851
insert_dft test_point 852
insert_dft user_test_point 858
insert_dft wir_signal_bits 860
insert_dft wrapper_cell 861
Command Reference for Encounter RTL Compiler

- **insert_dft**
  - wrapper_instruction_register 867
- **insert_dft**
  - wrapper_mode_decode_block 869
- **insert_test_compression** 871
- **insert_tiehilo_cells** 1077

**L**
- **lcd** 88
- **license** 89
- **license checkin** 90
- **license checkout** 91
- **license feature** 93
- **license list** 94
- **license version** 95
- **ll** 52
- **lls** 96
- **lpopd** 97
- **lpushd** 98
- **lpwd** 99
- **ls** 52, 53

**M**
- **man** 100
- **map_mbist_cgc_to_cgic** 874
- **merge_to_multibit_cells** 364
- **mesg_make** 1096
- **mesg_send** 1098
- **modify_power_domain_attr** 596
- **more** 101
- **move_blockage** 598
- **move_instance** 599
- **move_port** 600
- **move_region** 601
- **multi_cycle** 332
- **mv** 1080

**P**
- **parse_options** 1099
- **path_adjust** 336
- **path_delay** 340
- **path_disable** 343
- **path_group** 346
- **popd** 58
- **propagate_constraints** 349

- **pushd** 59
- **pwd** 60

**Q**
- **quit** 103

**R**
- **read_db** 210
- **read_def** 602
- **read_dfm** 212
- **read_dft_abstract_model** 875
- **read_encounter** 606
- **read_hdl** 216
- **read_ilm** 387
- **read_ilm_from_files** 388
- **read_io_speclist** 878
- **read_memory_view** 879
- **read_netlist** 222
- **read_pmbist_interface_files** 881
- **read_power_intent** 1032
- **read_saif** 992
- **read_sdc** 227
- **read_sdp_file** 607
- **read_spef** 608
- **read_tcf** 997
- **read_vcd** 1002
- **redirect** 104
- **remap_to_dedicated_clock_library** 365
- **remove_assigns_without_optimization** 36
- **remove_cdn_loop_breaker** 1082
- **remove_clock_reconvergence** 369
- **remove_inserted_sync_enable_logic** 370
- **replace_opcg_scan** 882
- **replace_scan** 884
- **report** 420
- **report area** 425
- **report boundary_opto** 427
- **report case_analysis** 429
- **report cdn_loop_breaker** 430
- **report cell_delay_calculation** 431
- **report clock_gating** 432
- **report clocks** 438
- **report congestion** 440
- **report datapath** 441
- **report design_rules** 446
- **report dft_chains** 448
report dft_clock_domain_info 453
report dft_core_wrapper 454
report dft_registers 458
report dft_setup 462
report dftViolations 466
report disabled_transparent_latches 469
report gates 470
report hierarchy 474
report instance 476
report low_power_cells 479
report memory 486
report memory_cells 487
report messages 488
report multibit_inferencing 491
report net_cap_calculation 496
report net_delay_calculation 497
report net_res_calculation 498
report nets 499
report opcg_equivalents 502
report ple 503
report port 505
report power 507
report power_domain 519
report proto 521
report qor 523
report scan_compressibility 529
report sequential 531
report slew_calculation 534
report summary 535
report test_power 537
report timing 541
report units 552
report yield 554
report_low_power_intent 482
reset_attribute 106
reset_design 1084
reset_opcg_equivalent 894
reset_scan_equivalent 895
resize_blockage 611
resize_region 612
restore_congestion_map 613
restore_design 232
resume 107
ret ime 371
rm 1061, 1085

S

save_congestion_map 614
sdc_shell 108

set_attribute 109
set_compatible_test_clocks 896
set_opcg_equivalent 898
set_remove_assign_options 373
set_scan_equivalent 900
sh 111
shell 112
specify_cell_pad 615
specify_floorplan 616
specify_paths 351
split_db 234
state_retention 1008
state_retention
connect_power_gating_pins 1009
state_retention_swap 1010
statistics 555
statistics add_metric 557
statistics log 558
statistics read 560
statistics remove_metric 561
statistics report 562
statistics reset 565
statistics run_stage_ids 566
statistics write 567
stop_suspend 113
string_representation 114
summary_table 618
suppress_messages 115
suspend 116
synthesize 377

t

test_super_thread_servers 120
timestat 568

U

ungroup 1086
unify 1088
unsuppress_messages 121
update_congestion_map 620
update_gcell_congestion 621
update_gcell_pin_density 622
update_gcell_utilization 623
update_scan_chains 902
Command Reference for Encounter RTL Compiler

V
validate_constraints 357
validate_timing 569
vdir_lsearch 61
verify_power_structure 1036
vname 63

W
what_is 64
what_is_list 65
write_atpg 904
write_bsv 907
write_compression_macro 910
write_db 238
write_def 624
write_design 241
write_dft_abstract_model 918
write_dft_rtl_model 921
write_do_ccd 245
write_do_ccd cdc 246
write_do_ccd compare_sdc 247
write_do_ccd generate 249
write_do_ccd validate 253
write_do_clp 254
write_do_lec 257
write_do_verify cdc 261
write_encounter 263
write_et_atpg 922, 923, 924, 948
write_et_bsv 930
write_et_dfa 934
write_et_lbist 938
write_et_mbist 942
write_et_no_tp_file 947
write_ets 273
write_ett 274
write_forward_saif 1011
write_hdl 276
write_io_speclist 952
write_ldb 281
write_logic_bist_macro 954
write_mbist_testbench 958
write_pmbist_interface_files 962
write_pmbist_testbench 964
write_power_intent 1038
write_saif 1013
write_scandef 968
write_script 289
write_sdc 292
write_sdf 296
write_sdp_file 625
write_set_load 301
write_spef 626
write_sv_wrapper 303
write_tcf 1015
write_template 307
Preface

- About This Manual on page 28
- Additional References on page 28
- How to Use the Documentation Set on page 29
- Reporting Problems or Errors in Manuals on page 30
- Customer Support on page 31
- Messages on page 32
- Man Pages on page 33
- Command-Line Help on page 34
- Documentation Conventions on page 36
About This Manual

This manual provides a concise reference of the commands available to the user when using Encounter™ RTL Compiler. This manual describes each command available within the RTL Compiler shell with its command options.

Additional References

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

- TCL Reference, Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley Publishing Company
- 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. 1364-2001)

Note: For information on purchasing IEEE specifications go to http://shop.ieee.org/store/ and click on Publications & Standards.
How to Use the Documentation Set

INSTALLATION AND CONFIGURATION

- Cadence Installation Guide
- Cadence License Manager
- README File

NEW FEATURES AND SOLUTIONS TO PROBLEMS

- README File
- What's New in Encounter RTL Compiler
- Known Problems and Solutions in Encounter RTL Compiler

TASKS AND CONCEPTS

- Getting Started with Encounter RTL Compiler
- Using Encounter RTL Compiler
- HDL Modeling in Encounter RTL Compiler
- ChipWare Developer in Encounter RTL Compiler
- Library Guide for Encounter RTL Compiler
- Datapath Synthesis in Encounter RTL Compiler
- Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler
- Low Power in Encounter RTL Compiler
- Design for Test in Encounter RTL Compiler

REFERENCES

- Attribute Reference for Encounter RTL Compiler
- Command Reference for Encounter RTL Compiler
- ChipWare in Encounter RTL Compiler
- GUI Guide for Encounter RTL Compiler
- Quick Reference for Encounter RTL Compiler
Reporting Problems or Errors in Manuals

The Cadence® Help online documentation, 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
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 RTL Compiler:

- Advanced Synthesis with Encounter RTL Compiler
- Basic Static Timing Analysis
- Encounter RTL Compiler
- Low-Power Synthesis Flow with Encounter RTL Compiler
- Test Synthesis Using Encounter RTL Compiler

For further information on the training courses available in your region, visit the Cadence Training portal. You can also write to training_enroll@cadence.com.

**Note:** The links in this section open in a new browser. They initially display the requested training information for North America, but if required, you can navigate to the courses available in other regions.

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

From within RTL Compiler there are two ways to get information about error messages.

- **Use the `report messages` command.**

  For example:

  ```
  rc:/> report messages
  ```

  This returns the detailed information for each message output in your current RTL Compiler run. It also includes a summary of how many times each message was issued.

- **Use the `man` command.**

  **Note:** You can only use the `man` command for messages within RTL Compiler.

  For example, to get more information about the “TIM-11” message, type the following command:

  ```
  rc:/> man TIM-11
  ```

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:

  ```
  rc_pubs@cadence.com
  ```
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 RTL Compiler. Man pages contain the
same content as the Command and Attribute References. To use the man pages from the
UNIX shell:

1. Set your environment to view the correct directory:
   setenv MANPATH $CDN_SYNTH_ROOT/share/synth/man

2. Enter the name of the command or attribute that you want either in RTL Compiler or
   within the UNIX shell. For example:
   - man check_dft_rules
   - man cell_leakage_power

You can also use the more command, which behaves like its UNIX counterpart. If the output
of a manpage is too small to be displayed completely on the screen, use the more command
to break up the output. Use the spacebar to page forward, like the UNIX more command.

rc:/> more man synthesize
Command-Line Help

You can get quick syntax help for commands and attributes at the RTL Compiler 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 this document does not necessarily match the information that you get when you type `help command_name`. 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:

rc_pubs@cadence.com

**Getting the Syntax for a Command**

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

For example:

```
rc:/> help path_delay
```

This returns the syntax for the `path_delay` command.

**Getting the Syntax for an Attribute**

Type the following:

```
rc:/> get_attribute attribute_name * -help
```

For example:

```
rc:/> get_attribute max_transition * -help
```

This returns the syntax for the `max_transition` attribute.
Searching for Attributes

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

rc:/> 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.

rc:/> set_attr li

ambiguous "li": lib_lef_consistency_check_enable lib_search_path libcell liberty_attributes libpin library library_domain line_number

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:

  rc:/> c <Tab>

  This returns the following commands:

  ambiguous "c": cache_vname calling_proc case catch cd cdsdoc change_names check_dft_rules chipware clear clock clock_gating clock_ports close cmdExpand command_is_complete concat configure_pad_dft connect_scan_chains continue cwd_install ..

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

  For example:

  rc:/> path_<Tab>

  This returns the following commands:

  ambiguous command name "path_": path_adjust path_delay path_disable path_group
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.
- RTL Compiler commands are shown following the `rc:`/`>` string.

Text Command Syntax

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

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

- basename on page 38
- cd on page 39
- dirname on page 41
- dirs on page 42
- filter on page 43
- find on page 46
- inout_mate on page 51
- ll on page 52
- ls on page 53
- popd on page 58
- pushd on page 59
- pwd on page 60
- vdir_lsearch on page 61
- vname on page 63
- what_is on page 64
- what_is_list on page 65
basename

basename pathname

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

Options and Arguments

pathname Specifies the 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:
  
  rc:/> basename \
  /hdl_libraries/CW/components/CW_absval/comp_architectures/CW_absval\n  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:
  
  rc:/> set libcell /libraries/LIB/libcells/ND2X1\n  rc:/> basename [dirname [dirname $libcell]]

Related Information

Related commands: dirname on page 41
cd

Sets the current directory in the design hierarchy and navigates the design hierarchy. This command is similar to its UNIX counterpart. A description of the design hierarchy is given in the Using Encounter RTL Compiler.

Options and Arguments

directory

Specifies the name of the directory to be set as the current directory.

The ".", "..", and "/" have the same meaning as their UNIX counterparts (current directory, parent directory, and root directory 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:
  
  ```
  rc:/designs> cd
  rc:/> pwd
  /
  ```

- The following command specifies the absolute path to the target directory:
  
  ```
  rc:/> cd /designs/alu/timing/exceptions
  ```

- The following command specifies the relative path to the target directory:
  
  ```
  rc:/> cd designs/alu
  rc:/designs/alu> cd timing/exceptions
  ```

- The following command changes the current directory to a parent directory:
  
  ```
  rc:/designs/alu/timing/exceptions cd ../../subdesigns
  rc:/designs/alu/subdesigns> pwd
  /designs/alu/subdesigns
  ```

- The following command uses wildcards in the path specification:
  
  ```
  rc:/designs/cmplx_alu/subdesigns/addinc> cd ../../../tim*/exc*
  rc:/designs/cmplx_alu/timing/exceptions> pwd
  /designs/cmplx_alu/timing/exceptions
  ```
Related Information

Affects these commands:  
dirs on page 42  
ls on page 53  
popd on page 58  
pushd on page 59  
pwd on page 60
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:
  
  ```
  rc:/> 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.
  
  ```
  rc:/> 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:
  
  ```
  rc:/> set libcell /libraries/LIB/libcells/ND2X1
  rc:/> basename [dirname [dirname $libcell]]
  ```

Related Information

Related commands: `basename` on page 38
dirs

Displays the contents of the design directory 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 directory stack, then display the contents of the directory stack:
  
  ```
  rc:/> pushd designs
  /designs /
  rc:/designs> pushd /libraries
  /libraries /designs /
  rc:/libraries> dirs
  /libraries /designs /
  ```

- The following commands respectively remove the last added directories and then display the contents of the directory stack:
  
  ```
  rc:/libraries> popd
  /designs /
  rc:/designs> popd
  /
  rc:/> dirs
  ```

Related Information

Related commands: ls on page 53
popd on page 58
pushd on page 59
pwd on page 60
filter

filter [-invert] [-special] [-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.

-special         Use this option when the attribute value contains special characters, such as square brackets.

-vname           Allows to specify the Verilog name instead of the RTL Compiler 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:

rc:/> filter preserve true [find . -libcell *]
The following command stores the Tcl list of all matching cells returned by the filter command in a variable for use in scripting later.

```
rc:/> 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.

```
rc:/> 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:

```
rc:/> 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:

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

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

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

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

Use the `-special` option to handle the special characters in the attribute value.

```
rc:/> set instances [find /designs/* -instance instances_seq/*] 
   [/designs/test/instances_seq/out_reg[3]] [/designs/test/instances_seq/out_reg[2]] 
   [/designs/test/instances_seq/out_reg[0]] 
rc:/> filter -special dft_test_clock 
   [/designs/test/dft/test_clock_domains/clk[0]/clk[0]] $instances 
   [/designs/test/instances_seq/out_reg[0]] [/designs/test/instances_seq/out_reg[1]]
```
The following command returns the design whose tns value equals 0.

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

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

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

### Related Information

Affects these commands:
- `ls` on page 53
- `get_attribute` on page 82
- `set_attribute` on page 109
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.

  Default: 0
Examples

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

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

- The following command finds all registers in all designs:

  rc:/> 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_3
  /designs/alu/instances_seq/aluout_reg_2 /designs/alu/instances_seq/aluout_reg_1

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

  rc:/> find des*/alu -port ports_in/*
The following command searches for an external delay whose name starts with in, starting from the designs directory:

```
rc:/designs> find designs -external_delay in*
/designs/alu/timing/external_delays/in_del_1
```

The following command finds all designs that are four characters:

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

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

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

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

```
rc:/> 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 RTL Compiler 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.

```
rc:/> find inst1 -pin out*
{/designs/woodward/instances_hier/inst1/pins_out/out1[3]}
rc:/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.

```
rc:/> find / -instance *
/designs/MOD69/instances_hier/inst1 /designs/MOD69/instances_hier/inst1/instances_comb/g21
```

The -vname option removes the container directories (in this case instance_hier) and presents the list more concisely:
rc:/> 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:
rc:/> find / -regexp (VLOGPT-6+) -messages * *

■ The following example uses the -invert option to return all library objects except for LIB2:
rc:> find / -invert "LIB2" -library *
/libraries/LIB1 /libraries/LIB3

■ The following example returns all combinational instances named g58 or g59 in the Verilog name style:
rc:/> find / -regexp {g[5][8-9]} -vname -instance instances_comb/*

■ Use the ls -dir command to format the output row over row:
rc:/> 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:
rc:/> 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.
rc:/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
rc:/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
rc:/designs> set rc [find / -split -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
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_1
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_0
rc:/designs> echo $rc
rc:/designs>

Related Information

Related command: clock_ports on page 409
Related attributes find_inefficient_threshold
find_inefficient_use
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 RTL Compiler 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:
    `rc:/> 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
    `rc:/> 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 RTL Compiler 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:
    `rc:/> 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:
    `rc:/> inout_mate s4/subports_out/io
    `/designs/top/instances_hier/s4/subports_in/io`
ll

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

Lists information about any objects in the design hierarchy in long format.

Alias for `ls -long`. 
**ls**

```
```

Lists information about any objects in the design hierarchy (designs, library cells, clocks, and so on). This command is similar to its UNIX counterpart.

**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 directory 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 directory.
- **object** Specifies the directory for which you want to list information.
  
  Default: current directory
- **-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 directory:
  
  ```
  rc:/> ls libraries
  /libraries:
  ./ em333s/ sm333s/
  ```
The following command shows information of cell `buf1` in regular and long listing format:

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

```
rc:/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:

```
rc:/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
```
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:

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

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

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

```
rc:/>ls -computed /designs/violet/instances_comb/U1/pins_in/S
timing_case_computed_value = 1
```
The following examples show the difference between the `ls -attribute` and `get_attribute` commands.

```
rc:/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
rc:/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`.

```
rc:/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/
rc:/designs/test/sdp_groups>
```

The following command lists the attributes that start with `p`.

```
rc:/ ls -a -regexp ^p+
Total: 9 items
.
./
  Attributes:
    peak_memory = 75.00 M bytes
    platform_wordsize = 64 bits
    print_error_info = true
    program_name = Encounter(R) RTL Compiler
    program_version = 14.20
```
Related Information

Related command: get_attribute on page 82
popd

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

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

**Examples**

- In the following example, the directory stack starts off as `/libraries /designs`: /libraries is the current directory and the top element of the directory 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 directory.

  rc:/libraries> dirs
  /libraries /designs
  rc:/libraries> popd
  /designs
  rc:/designs> dirs
  /designs
  rc:/designs> pwd
  /designs

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

  rc:/> cd /designs
  rc:/designs> dirs
  /designs
  rc:/designs> popd
  Directory stack empty
  /designs

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

**Related Information**

Affects these commands:  
dirs on page 42
ls on page 53
pushd on page 59
pwd on page 60
pushd

pushd directory

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

Options and Arguments

directory

Specifies the name of the directory to be set as target directory. 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 directory stack and the current directory is changed to it:

rc:/designs> pushd /libraries
/libraries /designs
rc:/libraries>

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

rc:/libraries> pushd ex*
Error : A single object was expected, but multiple objects were found.[TUI-62]
: The argument that found multiple objects was 'ex*'.
Usage: pushd <object>
  <object>:
   new target directory
Failed on pushd ex*

Related Information

Affects these commands: dirs on page 42
ls on page 53
popd on page 58
pwd on page 60
pwd

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

Examples

- The following example shows the current directory after changing the current directory first:

  rc:/> cd /designs
  rc:/designs> pwd
  /designs

Related Information

Related commands:
  dirs on page 42
  ls on page 53
  popd on page 58
  pushd on page 59
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.

rc:/> set pin [get_attr driver /designs/test/nets/oo]
/designs/test/instances_hier/mux_oo_5_10/pins_out/z
rc:/> 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
rc:/> vdir_lsearch $list $pin
-1
rc:/> 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
/designs/test/instances_comb/g4/pins_In/in_0 abc
rc:/> 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
rc:/> vdir_1search $list $pin
4
vname

vname [instance | net | 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
```
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`.

  ```
  rc:/> what_is clock2
  clock
  ```

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

  ```
  rc:/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*
  ```
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.

  rc:/> what_is_list CLK1
  exception cost_group

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

  rc:/> 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:

  rc:/> 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 64
General

- ? on page 69
- alias on page 70
- all_inputs on page 71
- all_outputs on page 72
- apropos on page 73
- attribute_exists on page 74
- clear on page 75
- clear_redline_terminal on page 76
- date on page 77
- enable_transparent_latches on page 78
- exec_embedded_script on page 79
- exit on page 81
- get_attribute on page 82
- get_liberty_attribute on page 84
- get_read_files on page 85
- help on page 86
- include on page 87
- lcd on page 88
- license on page 89
- license_checkin on page 90
- license_checkout on page 91
license feature on page 93
license list on page 94
license version on page 95
lls on page 96
lpopd on page 97
lpushd on page 98
lpwd on page 99
man on page 100
more on page 101
quit on page 103
redirect on page 104
reset_attribute on page 106
resume on page 107
sdc_shell on page 108
set_attribute on page 109
sh on page 111
shell on page 112
stop_suspend on page 113
string_representation on page 114
suppress_messages on page 115
suspend on page 116
tcl_load on page 117
test_super_thread_servers on page 120
unsuppress_messages on page 121
Provides help on the specified RTL Compiler 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 RTL Compiler commands.

- **file**
  
  Specifies the name of the file to which to write the help.

### Examples

- **The following example requests help for the**
  
  ```
  rc:/> ? 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
  ```

- **The following example requests help for the synthesize and report commands:**
  
  ```
  rc:/> ? synthesize report
  Commands are:
  
  Analysis
  report  generates one of various reports
  
  Synthesis
  synthesize  synthesizes the design
  ```
**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.

**Example**

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

```bash
rc:/> alias ai all_inputs
ai
rc:/> ai -h
    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
```
all_inputs

all_inputs [-design design]

Returns the input ports of the specified design.

Options and Arguments

- **-design design**
  Specifies the name of the top-level design for which you want to list all input ports.
  If you omit the design name, the input ports of all loaded designs are listed.

Examples

- The following example lists the input ports of all loaded designs.
  
  ```
  rc:/designs/top> all_inputs
  /designs/top/ports_in/enable {/designs/top/ports_in/in1[7]}
  {/designs/top/ports_in/in1[6]} {/designs/top/ports_in/in1[5]}
  {/designs/top/ports_in/in1[4]} {/designs/top/ports_in/in1[3]}
  {/designs/top/ports_in/in1[2]} {/designs/top/ports_in/in1[1]}
  {/designs/top/ports_in/in1[0]} {/designs/top/ports_in/in2[7]}
  {/designs/top/ports_in/in2[6]} {/designs/top/ports_in/in2[5]}
  {/designs/top/ports_in/in2[4]} {/designs/top/ports_in/in2[3]}
  {/designs/top/ports_in/in2[2]} {/designs/top/ports_in/in2[1]}
  {/designs/top/ports_in/in2[0]} /designs/top/ports_in/clk
  /designs/my_CG_MOD/ports_in/ck_in /designs/my_CG_MOD/ports_in/enable
  /designs/my_CG_MOD/ports_in/test /designs/my_CG_MOD_neg/ports_in/ck_in
  /designs/my_CG_MOD_neg/ports_in/enable /designs/my_CG_MOD_neg7ports_in/test
  ```

- The following example lists the input ports of design my_CG_MOD.
  
  ```
  rc:/designs/top> all_inputs -design my_CG_MOD
  /designs/my_CG_MOD/ports_in/ck_in /designs/my_CG_MOD/ports_in/enable
  /designs/my_CG_MOD/ports_in/test
  ```

- Use the `ls -dir` command to format the output of the command
  
  ```
  rc:/> ls -dir [all_inputs]
  /designs/ksable/ports_in/in1[0]
  /designs/ksable/ports_in/in1[1]
  /designs/ksable/ports_in/in1[2]
  ```

- Use the `ls -dir` command with the redirect arrow to redirect the output to a specified file:
  
  ```
  rc:/> ls -dir [all_inputs] > areid.txt
  ```

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

---

**Command Reference for Encounter RTL Compiler**

**General**

- **all_inputs**
  - **all_inputs [-design design]**
  
  Returns the input ports of the specified design.

**Options and Arguments**

- **-design design**
  
  Specifies the name of the top-level design for which you want to list all input ports.

  If you omit the design name, the input ports of all loaded designs are listed.

**Examples**

- The following example lists the input ports of all loaded designs.
  
  ```
  rc:/designs/top> all_inputs
  /designs/top/ports_in/enable {/designs/top/ports_in/in1[7]}
  {/designs/top/ports_in/in1[6]} {/designs/top/ports_in/in1[5]}
  {/designs/top/ports_in/in1[4]} {/designs/top/ports_in/in1[3]}
  {/designs/top/ports_in/in1[2]} {/designs/top/ports_in/in1[1]}
  {/designs/top/ports_in/in1[0]} {/designs/top/ports_in/in2[7]}
  {/designs/top/ports_in/in2[6]} {/designs/top/ports_in/in2[5]}
  {/designs/top/ports_in/in2[4]} {/designs/top/ports_in/in2[3]}
  {/designs/top/ports_in/in2[2]} {/designs/top/ports_in/in2[1]}
  {/designs/top/ports_in/in2[0]} /designs/top/ports_in/clk
  /designs/my_CG_MOD/ports_in/ck_in /designs/my_CG_MOD/ports_in/enable
  /designs/my_CG_MOD/ports_in/test /designs/my_CG_MOD_neg/ports_in/ck_in
  /designs/my_CG_MOD_neg/ports_in/enable /designs/my_CG_MOD_neg7ports_in/test
  ```

- The following example lists the input ports of design my_CG_MOD.
  
  ```
  rc:/designs/top> all_inputs -design my_CG_MOD
  /designs/my_CG_MOD/ports_in/ck_in /designs/my_CG_MOD/ports_in/enable
  /designs/my_CG_MOD/ports_in/test
  ```

- Use the `ls -dir` command to format the output of the command
  
  ```
  rc:/> ls -dir [all_inputs]
  /designs/ksable/ports_in/in1[0]
  /designs/ksable/ports_in/in1[1]
  /designs/ksable/ports_in/in1[2]
  ```

- Use the `ls -dir` command with the redirect arrow to redirect the output to a specified file:
  
  ```
  rc:/> ls -dir [all_inputs] > areid.txt
  ```

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

all_outputs [-design design]

Returns the output ports of the specified design.

Options and Arguments

- **-design design**  
  Specifies the name of the top-level design for which you want to list all output ports.
  
  If you omit the design name, the output ports of all loaded designs are listed.

Examples

- The following example lists the output ports of all loaded designs.
  
  ```
  rc:/designs/top> all_outputs
  
  /designs/top/ports_out/out1[1] /designs/top/ports_out/out1[0]
  /designs/top/ports_out/out2[1] /designs/top/ports_out/out2[0]
  
  /designs/my_CG_MOD/ports_out/ck_out /designs/my_CG_MOD_neg/ports_out/ck_out
  ```

- The following example lists the output ports of design **my_CG_MOD**.
  
  ```
  rc:/designs/top> all_outputs -design my_CG_MOD
  
  /designs/my_CG_MOD/ports_out/ck_out
  ```

- Use the `ls -dir` command to format the output of the command
  
  ```
  rc:/> ls -dir [all_inputs]
  
  /designs/ksable/ports_out/out1[0]
  /designs/ksable/ports_out/out1[1]
  /designs/ksable/ports_out/out1[2]
  ```

- Use the `ls -dir` command with the redirect arrow to redirect the output to a specified file:
  
  ```
  rc:/> ls -dir [all_outputs] > areid.txt
  ```
  You can also append arrows ("\>>").
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_command` 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`:
  
  rc:/> apropos generic
  Commands with option text matching search string:
  - `synthesize`
  - `write_hdl`
  
  Attributes with help text matching search string:
  - `dp_perform_csa_operations (root)`
  - `dp_perform_sharing_operations (root)`
  - `dp_perform_speculation_operations (root)`
  - `hdl_auto_sync_set_reset (root)`
  - `timing_driven_muxopto (design)`
  - `timing_driven_muxopto (subdesign)`

- The following example searches for the term `generic` among commands only:
  
  rc:/> apropos -skip_attributes generic
  Commands with option text matching search string:
  - `synthesize`
  - `write_hdl`
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.
  
  rc:/> attribute_exists osc_source -type instance 0

- The following command checks whether count exists for message TUI-20.
  
  rc:/> attribute_exists count -path /messages/TUI/TUI-20 1
clear

clear

Clears the terminal screen.
clear_redline_terminal

clear_redline_terminal

Clears the terminal screen.
date

date

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

Examples

rc:/> date
Thu Apr 23 01:28:55 PM PDT 2009
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 469
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.

■ If both a design and a subdesign is specified, then the subdesign setting is ignored.
Related Information

Related command:    elaborate on page 360
Related attributes: hdl_auto_exec_sdc_scripts
exit

exit [code]

Exits from the RTL Compiler 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
get_attribute

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

Retrieves the value of an attribute set by RTL Compiler 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:
  
  rc:/> 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:
  
  rc:/> get_attribute library
tutorial.lbr
The following example stores the attribute value in a variable:

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

The following example returns the area of library cell:

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

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

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

The following example lists all attributes for all object types:

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

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

- **Using the `ls -attribute` command:**

```
rc:/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` command:**

```
rc:/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 109

**Related command:** `ls` on page 53
get_liberty_attribute

get_liberty_attribute attribute_name
   {libarc|libcell|libpin|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|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.

rc:/> get_liberty_attribute default_operating_conditions tutorial typical_case
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

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

help [command]...[> file]

Provides help on the specified RTL Compiler 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 RTL Compiler commands.

file Specifies the name of the file to which to write the help.

Examples

- The following example requests help for the report command:
  
  rc:/> help report
  That command is:
    report    generate one of various reports

- The following example requests help for the synthesize and report commands:
  
  rc:/> help synthesize report
  Commands are:
    report    generate one of various reports
    synthesize   synthesize the design
include
include file

Executes scripts containing RTL Compiler or Tcl commands in the order they are listed in the include command.

When RTL Compiler begins executing, a number of scripts are automatically included. Information on these configuration scripts is given in the Using Encounter RTL Compiler.

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:
  rc:/> include constraint.g
- The following example includes multiple scripts at once:
  rc:/> include constraint.g; include synth.g

Related Information

Affected by these attributes: hdl_search_path
lib_search_path
script_search_path
**Icd**

`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.
  
  `rc:/> lcd rtl_lab01`

**Related Information**

- Affects these commands:  
  - `lls` on page 96
  - `lpwd` on page 99
  - `shell` on page 112
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 90
license checkout on page 91
license feature on page 93
license list on page 94
license version on page 95
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:

- RTL_Compiler_CPU_Accel_Option
- RTL_Compiler_Ultra
- RTL_Compiler_Ultra_II_Option

Examples

- The following example checks in the RTL_Compiler_Ultra license:
  
  rc://> license checkin RTL_Compiler_Ultra
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 RTL Compiler.

Options and Arguments

license Specifies the license to be checked out. The licenses that can be checked out are:

- RC_NG100
- RTL.Compiler_Adv_Phys_Option
- RTL.Compiler_CPU_Accel_Option
- RTL.Compiler_Low_Power_Option
- RTL.Compiler_Ultra
- RTL.Compiler_Ultra_II_Option

-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 RTL Compiler. The default wait timeout is 600s.

Note: This option does not require the -queue option to be specified when starting RTL Compiler.

Examples

The following example checks-out the RTL.Compiler_Ultra license:

```
rc:/> license checkout RTL.Compiler_Ultra
Checking out license 'RTL.Compiler_Ultra'......   (1 second elapsed)
1
```
The following command checks out version 8.2 of the RTL_Compiler_Ultra_II_Option license.

```
license checkout RTL_Compiler_Ultra_II_Option -version 8.2
```
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:

- RC_NG100
- RTL_Compiler_Adv_Phys_Option
- RTL_Compiler_CPU_Accel_Option
- RTL_Compiler_Low_Power_Option
- RTL_Compiler_Ultra
- RTL_Compiler_Ultra_II_Option

-version float

Specifies the license version.

Examples

- The following example checks if the RTL_Compiler_Ultra product license exist:

  rc:/ license feature RTL_Compiler_Ultra

  The returned value “1” indicates that the license is available.
license list

license list

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

Example

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

```bash
rc:/> license list
RTL.Compiler.Low.Power.Option
```
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:

- RTL_Compiler_Adv_Phys_Option
- RTL_Compiler_CPU_Accel_Option
- RTL_Compiler_Low_Power_Option
- RTL_Compiler_Ultra
- RTL_Compiler_Ultra_II_Option
lls

lls directory

Lists the contents of the specified UNIX directory.

Options and Arguments

directory

Specifies the UNIX directory whose contents to list.

Examples

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

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

Related Information

Affected by this command: `lcd` on page 88

Related commands: `lpwd` on page 99, `shell` on page 112
**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.

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

**Related Information**

Affects these commands:  
- `lls` on page 96  
- `lpushd` on page 98  
- `lpwd` on page 99
Ipushd

Ipushd 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.

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

**Related Information**

Affects these commands:  
- `lils` on page 96  
- `lpopd` on page 97  
- `lpwd` on page 99
Ipwd

lpwd

Returns the UNIX working directory.

Examples

- The following example shows the current UNIX directory.
  
rccd:/> lpwd
/usr3/verilog_labs

Related Information

Affected by this command: lcd on page 88
Related commands: lls on page 96
                  shell on page 112
man

man \{ command_name \| attribute_name \| message_ID\}

Returns detailed information about the specified command, attribute, or message from the online reference manual. You must first set your MANPATH environment variable to the following path:

$CDN_SYNTH_ROOT/share/synth/man

After setting the MANPATH variable, you can obtain manpages for commands and attributes both within RTL Compiler or within the Unix shell. To obtain more information about RTL Compiler messages, you must use the man command within RTL Compiler.

Example

■ The following example returns the manpage for the synthesize command:

rc:/> man synthesize
User Commands synthesize(1)

NAME
synthesize

SYNTAX
synthesize \[-effort \{high\|medium\|low\}\] \[-to_generic\]
\[-to_mapped\] \[-incremental\] \langle design\rangle...

DESCRIPTION
Determines the most suitable design implementation using the given design constraints (clock cycle, input delays, output delays, technology library, and so on).

■ The following example returns information about the TIM-11 message:

rc:/> man TIM-11
Entry : TIM-11
Severity : Warning
Verbosity : Message is visible at any 'information_level' above '1'.
Description : Possible timing problems have been detected in this design.
Help : Use 'report timing -lint' for more information
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.

rc:/> 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
;
- 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
quit

Exits from the RTL Compiler 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
redirect


Redirects the standard output to a file or variable. You can write out a gzip compressed file by adding the .gz extension to the filename.

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.

Options and Arguments

-append Append the generated output to the specified file or Tcl variable.
-mesg Prevents messages from being suppressed.
-tee Also writes the output to standard output (stdout).
-variable Redirects the output to a Tcl variable.
command Specifies the command or Tcl code to execute as a quoted string.
target Specifies the name of the file or variable to which to write the output.

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.

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

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

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

```text```
reset_attribute

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

Resets an attribute to its default value.

You can also use this command to list the attributes whose value can be reset.

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:
  rc:/> reset_attribute retime* *

■ The following example specifies that all attributes on all sequential instances be returned to their default values:
  rc:/> reset_attribute * [find / -instance *seq/*]

■ The following command lists all valid attributes that you can reset:
  rc:/> reset_attribute -h * *

  Both type and attribute_name accept wildcard strings.
resume

Restarts the current Tcl process and brings back the regular rc:/> 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 113  
  * suspend on page 116
sdc_shell

Open the SDC shell within RTL Compiler. All SDC commands can be used without the dc:: prefix inside the SDC shell. The command exit quits the SDC shell and returns you back to the RTL Compiler prompt.

Example

Example

- The following example launches the SDC shell within RTL Compiler and then exits:

  ```
rcre/> sdc_shell
  Info  : Entering sdc_shell. [SDC-300]
  : All sdc commands will work without the dc:: prefix inside sdc_shell.
  Type ‘exit’ to leave the shell.
  sdc_shell> exit
  Info  : Leaving sdc_shell. [SDC-301]
  : Type sdc_shell to use it again.
  rc:/> 
```
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 RTL Compiler. 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 Attribute Reference for Encounter RTL Compiler.

Note: Not all attributes can be set. Attempting to set a read-only attribute returns an error. The Attribute Reference for Encounter RTL Compiler 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.
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:
  
  ```
  rc:/> set_attribute * * -help
  ```

  Both type and attribute_name accept wildcard strings.

- The following example lists all valid attribute names that contain the string dont:
  
  ```
  rc:/> 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:
  
  ```
  rc:/> 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:
  
  ```
  rc:/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 lib_search_path attribute. For the rest of the session, the lib_search_path attribute becomes read-only:
  
  ```
  rc:/> set_attribute -lock lib_search_path /home/Test/foo
  ```

Related Information

Affects these commands: ls on page 53
get_attribute on page 82
sh

sh command_string

Executes a UNIX shell command from the RTL Compiler shell.

When executing shell commands, RTL Compiler uses /bin/sh.

⚠️ Important

Attempting to change the working directory for the RTL Compiler 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:
  
  `rc:/> sh date
  ...

...`
shell

shell command_string

Executes a UNIX shell command from the RTL Compiler shell.

When executing shell commands, RTL Compiler uses /bin/sh.

⚠️ Important

Attempting to change the working directory for the RTL Compiler 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:

  `rc:/> shell date
  ...

Related Information

Affected by this command: `lcd` on page 88

Related commands: `lls` on page 96

`lpwd` on page 99
stop_suspend

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

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

Related Information

Related commands: 

- resume on page 107
- suspend on page 116
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.

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

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

suppress_messages [-n integer] [-quiet] message_id...

Suppresses printing of the specified message in the log file.

Options and Arguments

message_id  
Specifies the message identification.

-n integer  
Prints the specified message only \(n\) number of times. 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:

```
rc:/> 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 '/': '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 '/': 'lib_search_path' = /home/rcap/regs/rc/dft/SMG/native/analysis/test2/LIB

Related Information

Related command: unsuppress_messages on page 121
suspend

suspend

Stops the current Tcl process and brings up the rc-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.

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 RTL Compiler’s command editor history.

Related Information

Related commands: resume on page 107
stop_suspend on page 113
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 RTL Compiler 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.
The initialization procedure must match the following prototype:

```c
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 `fileName`es, 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).
Example

The following is a minimal extension:

```c
#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:

```bash
# 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
test_super_thread_servers

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

Example

rc:/> set_attr super_thread_servers {rcl117 rcl118} /
    Setting attribute of root '/': 'super_thread_servers' = rcl117 rcl118
rc:/> 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'.


unsuppress_messages

unsuppress_messages message_id...

Allows a previously suppressed message to be printed.

Options and Arguments

message_id Specifies the message identification.

Examples

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

rc:/> suppress_messages { VLOG-1 VLOG-2 }
rc:/> unsuppress_messages { VLOG-1 VLOG-2 }

Related Information

Related command: suppress_messages on page 115
GUI Text

- General GUI Text Commands on page 124
- HDL Viewer GUI Text Commands on page 130
- Schematic Viewer GUI Text Commands on page 133
- Physical Viewer GUI Text Commands on page 138
General GUI Text Commands

- `gui_balloon_info` on page 125
- `gui_hide` on page 125
- `gui_info` on page 125
- `gui_legend` on page 126
- `gui_raise` on page 127
- `gui_reset` on page 127
- `gui_resume` on page 128
- `gui_selection` on page 128
- `gui_show` on page 128
- `gui_status` on page 129
- `gui_suspend` on page 129
- `gui_update` on page 129
**gui_balloon_info**

gui_balloon_info

Retrieves the text from the last balloon info.

**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 127
- `gui_show` on page 128

**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).
gui_legend

 gui_legend -title *title* [-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** *title* 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 TCI 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.
Figure 3-1 GUI Legend

![GUI Legend Example]

**gui_raise**

`gui_raise [-nosync]`

Keeps the GUI window on top of all other windows.

**Options and Arguments**

- `-nosync` 

  Does not synchronize the GUI when the GUI is raised.

**Related Information**

Related commands:  
- `gui_hide` on page 125
- `gui_show` on page 128

**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 not get reset or cleared.
gui_resume

Resumes automatic GUI updates.

Related Information

Related command: gui_suspend on page 129

gui_selection

Returns the selection list, which consists of the object path for instance, net, pin, and port objects. The list can contain multiple objects.

Example

- The following example returns a combinational instance:

  rc:/> 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.

gui_show

Displays the GUI after using gui_hide command.

Options and Arguments

  -nosync  Does not synchronize the GUI when the GUI is displayed.

Related Information

Related commands: gui_hide on page 125
  gui_raise on page 127
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).

gui_suspend

gui_suspend

Suspends the automatic GUI updates.

Related Information

Related command: gui_resume on page 128

gui_update

gui_update

Updates the GUI. This is the same as selecting Update GUI from the File menu.

Related Information

Related commands: gui_resume on page 128
gui_suspend on page 129
HDL Viewer GUI Text Commands

- `gui_hv_clear` on page 131
- `gui_hv_get_file` on page 131
- `gui_hv_load_file` on page 131
- `gui_hv_set_indicators` on page 132
gui_hv_clear

gui_hv_clear
Removes all the data in the HDL Viewer.

gui_hv_get_file

gui_hv_get_file
Returns the name of the file currently loaded in HDL Viewer.

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.
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.

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 3-2.
  
  ```
  rc:/> gui_hv_set_indicators 12 10
  ```

- The following commands sets the line number to 12, the column number to 10, and removes the highlighting:
  
  ```
  rc:/> gui_hv_set_indicators 12 10 -clear
  ```

Figure 3-2  Setting Line and Column Numbers in the HDL Viewer
Schematic Viewer GUI Text Commands

- gui_sv_clear on page 134
- gui_sv_cone on page 134
- gui_sv_get_instance on page 134
- gui_sv_grey on page 134
- gui_sv_highlight on page 135
- gui_sv_load on page 136
- gui_sv_snapshot on page 137
- gui_sv_toolbar_button on page 137
gui_sv_clear

 gui_sv_clear

Clears highlighting and selection lists in the Schematic Viewer.

gui_sv_cone

 gui_sv_cone 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

 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).

 gui_sv_get_instance

 gui_sv_get_instance

Returns the objects for the currently displayed hierarchical instance.

 gui_sv_grey

 gui_sv_grey [on | off]

Controls the grey mode. You can also right-click the mouse button 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.
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       Appends an object to the highlighted list.
-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

```tcl
# 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
        }
    }
}
```
- The following example highlights the falling clock edge flip-flops

```tcl
# 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**

```tcl
gui_sv_load {design | instance}
```

Specifies an hierarchical instance or design to load into the main Schematic Viewer.
gui_sv_snapshot

gui_sv_snapshot
    [-overwrite] [-no_fit] [-ps] file

Saves a snapshot 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            Specifies to not fit the snapshot to the full screen.
    -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 this option is omitted, 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.
                        The procedure must be loaded before you press the button to
                        call it.
    -tip string        Specifies the tool tip to be displayed.
Physical Viewer GUI Text Commands

- gui_pv_airline_add on page 140
- gui_pv_airline_add_custom on page 142
- gui_pv_airline_delete on page 143
- gui_pv_airline_display on page 144
- gui_pv_airline_raw_add on page 145
- gui_pv_airline_raw_add_custom on page 146
- gui_pv_align_instance_to_boundary on page 147
- gui_pv_align_instances on page 148
- gui_pv_clear on page 148
- gui_pv_connectivity_airlines on page 149
- gui_pv_deselect on page 150
- gui_pv_display_collection on page 151
- gui_pv_draw_box on page 152
- gui_pv_draw_circle on page 153
- gui_pv_draw_line on page 154
- gui_pv_draw_triangle on page 155
- gui_pv_get_design on page 156
- gui_pv_grey on page 156
- gui_pv_highlight on page 157
- gui_pv_highlight_hier_instances on page 159
- gui_pv_highlight_update on page 160
- gui_pv_label on page 161
- gui_pv_lineup_instances on page 162
- gui_pv_new_viewer on page 162
- gui_pv_preferences on page 163
- gui_pv_redraw on page 164
gui_pv_select on page 164
gui_pv_selection on page 165
gui_pv_snapshot on page 166
gui_pv_steiner_tree on page 168
gui_pv_toolbar_button on page 169
gui_pv_update on page 170
gui_pv_zoom_box on page 171
gui_pv_zoom_fit on page 171
gui_pv_zoom_in on page 172
gui_pv_zoom_out on page 172
gui_pv_zoom_to on page 173
gui_pv_airline_add

gui_pv_airline_add -name name [-id integer]
   -from {port|instance} -to {port|instance}
   [-color color] [-label label] [-nodisplay] [-width integer]

Adds the specified airline to the Physical Viewer.

Options and Arguments

- **-color color** Specifies the airline color.
  
  Default: blue

- **-from {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.

- **-to {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.

Example

- The following command creates a red airline named test from the g442 port to the g412 port with the airline label, as shown in Figure 3-3.

  gui_pv_airline_add -from g442 -to g412 -color red -label airline -name test
Figure 3-3  Specified Airline in the Physical Viewer

Related Information

Related command:  gui_pv_new_viewer on page 162
gui_pv_airline_add_custom

gui_pv_airline_add_custom -name name [-id integer]
   -from {port|instance} -to {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.

Options and Arguments

- **-color color**
  Specifies the airline color.
  
  *Default: blue*

- **-from {instance | port}**
  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 {instance | port}**
  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.
Related Information

Related command: gui_pv_new_viewer on page 162

**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.

- **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:
  
  rc:/> gui_pv_airline_delete test

**Related Information**

Related command: gui_pv_new_viewer on page 162
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.

- **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 removes the airline test from the Physical Viewer:
  ```
  rc:/> gui_pv_airline_display test
  ```

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_airline_raw_add

gui_pv_airline_raw_add [-name name] [-id integer]
   [-color color] [-label label] [-width integer]
   [-nodisplay] -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.
   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.
-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.

Related Information

Related command: gui_pv_new_viewer on page 162
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.

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.
Related Information

Related command: gui_pv_new_viewer on page 162

**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.

**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.

Options and Arguments

-bottom         Aligns the bottom sides of the specified instances.
instance        Specifies the instances to be aligned.
-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] [-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.
-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.
-selection      Deselects all objects.

Related Information

Related command: gui_pv_new_viewer on page 162
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.

- **{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 162
gui_pv_deselect

gui_pv_deselect object... [-id integer]

Specifies the list of object types to deselect in the physical viewer.

You can specify any of the following:

- blackbox
- cluster
- instance
- pin
- region
- blockage
- def_pin
- pcell
- pinstance
- row
- bump
- gcell
- pdomain
- port

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 162
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.

Related Information

Related commands:  
gui_pv_draw_box on page 152  
gui_pv_draw_circle on page 153  
gui_pv_draw_line on page 154  
gui_pv_draw_triangle on page 155  
gui_pv_highlight on page 157  
gui_pv_new_viewer on page 162
**gui_pv_draw_box**

```bash
gui_pv_draw_box -llx float -lly float
    -width float -height float
    [-color color] [-collect] [-append] [-stipple]
    [-group name] [-label label] [-dbu] [-id integer]
```

Draws a box on top of the physical view. You can use this command to annotate information.

**Options and Arguments**

- **-append** Append the object to highlight.
- **-collect** Collects objects to highlight.
- **-color color** Specifies the highlight color.
  
  **Default:** red
- **-dbu** Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
- **-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 box.
- **-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.
- **-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.
- **-label string** Specifies the object label.
- **-stipple** Specifies to use stipple fill pattern to highlight the object.
- **-width float** Specifies the width of the box in micrometer.

**Related Information**

Related command: `gui_pv_new_viewer` on page 162
gui_pv_draw_circle

gui_pv_draw_circle -cx float -cy float
   -radius float
   [-color color] [-collect] [-append] [-stipple]
   [-group name] [-label label] [-dbu] [-id integer]

Draws a circle on top of the physical view. You can use this command to annotate information.

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.
-db u                      Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
-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 box.
-group name                Displays only those objects in the specified group. These objects were previously marked for display through the gui_pv_highlight -collect command.
-label string              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.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_draw_line

gui_pv_draw_line -fx float -fy float
   -tx float -ty float
   [-color color] [-collect] [-append] [-stipple]
   [-group name] [-label label] [-dbu] [-id integer]

Draws a line between the specified coordinates on top of the physical view. You can use this command to annotate information.

Options and Arguments

- **-append**
  Appends the object to highlight.

- **-collect**
  Collects objects to highlight.

- **-color color**
  Specifies the highlight color.
  
  *Default:* red

- **-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.

*Note:* One DB unit is 1/1000 micron.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_draw_triangle

gui_pv_draw_triangle -llx float -lly float
    -width float -height float
    [-color color] [-collect] [-append] [-stipple]
    [-group name] [-label label] [-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.

Options and Arguments

-append Appends the object to highlight.
-collect Collects objects to highlight.
-color color Specifies the highlight color.
    Default: red
-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.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_get_design

gui_pv_get_design [-id integer]

Returns the current design displayed in the specified 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.

Related Information

Related command: gui_pv_new_viewer on page 162

gui_pv_grey

gui_pv_grey [on | off] [-id integer]

Controls the grey mode. You can also right-click the mouse button in the Schematic Viewer and select Grey Mode On or Grey Mode Off.

Options and Arguments

-id integer In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

-on Turns on grey mode in the Schematic Viewer.

-off Turns off grey mode in the Schematic Viewer.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_highlight

gui_pv_highlight [-color color] [-collect] [-append]
    [-group name] [-label string] [-stipple]
    {blockage | gcell | instance | pcell | pdomain | pin | port | region | row}...[-id integer]

Highlights objects in the Physical Viewer

Options and Arguments

- **-append** Appends the object to highlight.
- **-collect** Collects objects to highlight
- **-color color** Specifies the color for highlighting.
  Default: red
- **{blockage | gcell | instance | pcell | pdomain | pin | port | region | row}** Specifies the object to highlight if it is in the scope of the current Physical Viewer.
- **-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.
- **-label string** Specifies the object label.
- **-stipple** Specifies to use stipple fill pattern to highlight the object.

Examples

- The following command highlights the g411 port in yellow (see Figure 3-4 on page 158):
  rc:// gui_pv_highlight -color yellow g411

- The following command adds the g405 port to the highlighted list (see Figure 3-4 on page 158):
  rc:// gui_pv_highlight -color yellow -append g405
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:

```
rc:/> gui_pv_highlight -collect /designs/bree/instances_hier/eve1/ \
    -group laurence
```

Related Information

Related commands: gui_pv_clear on page 148
gui_pv_new_viewer on page 162
gui_pv_highlight_hier_instances

gui_pv_highlight_hier_instance
    [-depth integer] [-under instance]
    [-instances instance_list] [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.
    -instances instance_list Specifies the instances to be highlighted.
    -under instance         Specifies the parent hierarchical instance that contains the instances to be highlighted.
gui_pv_highlight_update

gui_pv_highlight_update -property string [-id integer]
   -value string [-group string]
   {blockage | gcell | instance | pcell | port | region}...

Updates the object highlight in the Physical Viewer.

Options and Arguments

{blockage|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: guipvniewer on page 162

// test1.e
gui_pv_label

gui_pv_label
  [-color color] -x float -y float label [-id integer]

Adds a text label at the specified point in the Physical Viewer.

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 162
gui_pv_lineup_instances

gui_pv_lineup_instances -space float
  [-top] [-bottom] [-left] [-right]
  [instance]...

Evenly distributes the instances vertically or horizontally starting from the specified side.

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.

This argument is optional if there is only one design in memory.
gui_pv_preferences

gui_pv_preferences [-name] [-selectable] [-visible] [-color string] [-style {outline | fill | stipple}] string [-id integer]

Configures the preferences for the specified object type in the Physical Viewer.

**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**
  Specifies to display the name of the objects of the specified type.

- **-selectable**
  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**
  Specifies the drawing style for the object type.

  *Default:* outline.

- **-visible**
  Specifies that the objects of the specified type must be visible.

**Related Information**

Related command: gui_pv_new_viewer on page 162
**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.

**Related Information**

Related command: **gui_pv_new_viewer** on page 162

---

**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
- cluster
- instance
- pin
- region
- blockage
- def_pin
- pcell
- pininstance
- row
- bump
- gcell
- pdomain
- port

**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 162
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.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_snapshot

gui_pv_snapshot
    [-congestion] [-pin_density] [-utilization]
    [-no_fit] [-overwrite] [-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.

-no_fit Specifies to not fit the snapshot to the full screen.

-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.

-ps Specifies to create the file in PostScript format.
    
    If this option is omitted, a GIF file will be created.

-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.
Examples

- The following command saves a snapshot of the utilization map in the `util.gif` file.
  
  ```
  gui_pv_snapshot -utilization util.gif
  ```

- 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 162

**gui_pv_steiner_tree**

```bash
gui_pv_steiner_tree
   [-append] {instance | port }... [-id integer]
```

Shows the steiner tree from the output pins of the specified instance or from the specified port.

**Options and Arguments**

- **-append**
  Adds the new steiner trees 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.

- **{instance | 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 162
**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.

**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.

- **-id integer**
  In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

- **-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.

**Related Information**

Related command:  
[gui_pv_new_viewer](#) on page 162
gui_pv_update

gui_pv_update [string] [-design design] [-id integer]

Specifies the objects to be updated in the Physical Viewer.

Options and Arguments

- **-design design**
  Specifies the design for which to update the GUI.

- **-id integer**
  In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

- **string**
  Specifies the objects to be updated in the Physical Viewer. You can specify row or region.
  By default, all objects will be updated.

Example

In the following example, the first two command create a new row without GUI update. The third command request a GUI update to show the new rows.

```
create_row -no_update ...
create_row -no_update ...
gui_pv_update row
```

Related Information

Related command: gui_pv_new_viewer on page 162
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.

- **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 162

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.

Related Information

Related command: gui_pv_new_viewer on page 162
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.

Related Information

Related command: gui_pv_new_viewer on page 162

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.

Related Information

Related command: gui_pv_new_viewer on page 162
gui_pv_zoom_to

gui_pv_zoom_to [-id integer]

Zooms to the bounding box around selected objects 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.

Related Information

Related command: gui_pv_new_viewer on page 162
Chipware Developer

- `cwd` on page 176
- `cwd check` on page 177
- `cwd create_check` on page 181
- `cwd report_check` on page 183
- `hdl_create` on page 185
- `hdl_create binding` on page 186
- `hdl_create component` on page 188
- `hdl_create implementation` on page 190
- `hdl_create library` on page 192
- `hdl_create operator` on page 193
- `hdl_create package` on page 194
- `hdl_create parameter` on page 196
- `hdl_create pin` on page 198
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 177
                  cwd create_check on page 181
                  cwd report_check on page 183
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_lib
- hdl_oper
- hdl_comp
- hdl_bind
- hdl_impl
- hdl_param
- hdl_pin
- hdl_pack
- hdl_arch

The RTL Compiler path to the hdl_objects to be checked can either be an absolute path:
rc:/> cwd check /hdl_libraries/CW/components/CW_add

Or it can be a relative path with respect to the current working directory:
rc:/> cd /hdl_libraries/CW/components
rc:/> cwd check CW_add

You can use wild cards for specifying multiple hdl_objects:
rc:/> cwd check /hdl_libraries/CW/components/*add*

or:
rc:/> cd /hdl_libraries/CW/components
rc:/> cwd check *add*
You can specify multiple directories at the same time:

```
rc:/> cwd check {/hdl_libraries/CW /hdl_libraries/DW02}
```

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.

- `-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.

```
rc:/> get_attribute information_level
1
rc:/> 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
nonbuiltin 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
```

-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.

```
rc:/> get_attribute information_level
1
rc:/> 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
nonbuiltin 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
```
The following example runs checking rules on all the `hdl_objects` under `parameters` and produces a verbose report:

```
rc:/> 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

<table>
<thead>
<tr>
<th>Check_name</th>
<th>Passed</th>
<th>Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter_formula</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
```

The following example will check the `CW_add` component, but will skip checking its bindings and implementations:

```
rc:/> cd /hdl_libraries/CW/components/CW_add
rc:/> cwd check . -skip { /hdl_libraries/CW/components/CW_add/bindings/* \ /hdl_libraries/CW/components/CW_add/implementations/* }
```

Related Information

**Checking Rules in ChipWare Developer**
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.

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 RTL Compiler. 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|high}  Specifies the effort level.

Default: low

  low — The check is not allowed to parse the HDL code of the synthesis models.
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.

```
rc> cwd create_check -check_name "arch_pin_order" -effort "medium" \
   -severity 0 -description "Check Whether the order of pins specified \n   in the synthesis model is consistent with what is defined in the \n   registration script" \
   -checklist { (HDL_ARCH_PINS_SCANNED check_arch_pin_order) \n   {HDL_COMP_PINS_SCANNED check_component_pin_order} }
```

Related Information

**Checking Rules** in *ChipWare Developer*
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 both used simultaneously.

Options and Arguments

-checkname 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.
Examples

- The following example reports details about the checking rule `arch_pin_order`, which uses two check procs to associate with two checkpoints.

```bash
rc:/> 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.

<table>
<thead>
<tr>
<th>Check_name</th>
<th>Checkpoint</th>
<th>Check_proc</th>
<th>Eff</th>
<th>Seve</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch_pin_order</td>
<td>HDL_ARCH_PIN ::cwd::arch_pin</td>
<td>med</td>
<td>0</td>
<td></td>
<td>Check whether the order of pins specified in the synthesis model is consistent with what is defined in the registration script</td>
</tr>
<tr>
<td></td>
<td>S_SCANNED</td>
<td>::cwd::pro</td>
<td>ium</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HDL_COMP_PIN ::cwd::componen</td>
<td>t_pin_order::ch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The following example reports details about the checkpoint `HDL_OPER_BINDINGS_SCANNED`:

```bash
rc:/> cwd report_check -checkpoint {HDL_OPER_BINDINGS_SCANNED} -max_width {Check_name 10} {Checkpoint 15} {Check_proc 15} {Effort 3} {Severity 4} {Description 20}
```

<table>
<thead>
<tr>
<th>Check_name</th>
<th>Checkpoint</th>
<th>Check_proc</th>
<th>Eff</th>
<th>Seve</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator_b</td>
<td>HDL_OPER_BINDIN ::cwd::operator</td>
<td>low</td>
<td>1</td>
<td></td>
<td>check that for every hdl_bindings defined for the hdl_operator there is at least one attribute is set to false</td>
</tr>
<tr>
<td></td>
<td>GS_SCANNED</td>
<td>::cwd::chec k_proc</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The following example reports all checking rules that have been registered:

```bash
rc:/> cwd report_check -checkname {*}
```

- This reports info about all checkpoints:

```bash
rc:/> cwd report_check -checkpoint {*}
```

- The following example reports all checking rules whose checkname contains string "pin":

```bash
rc:/> cwd report_check -checkname {*pin*}
```

- The following example reports information about the `arch_pin_order` and `arch_parameter_order` checking rules:

```bash
rc:/> cwd report_check -checkname {arch_pin_order arch_parameter_order}
```
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 186
hdl_create component on page 188
hdl_create implementation on page 190
hdl_create library on page 192
hdl_create operator on page 193
hdl_create package on page 194
hdl_create parameter on page 196
hdl_create pin on page 198
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

You can save run-time if you cd to the component name directory and issue 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.

    rc:/hdl_libraries/my_CW/components/my_CW_mult> hdl_create binding \
    test1 -operator MY_MULT_OP
    rc:/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.

    rc:/> hdl_create binding test1 -operator MY_MULT_OP \
    /hdlLibraries/my_CW/components/my_CW_mult
    rc:/> ls /hdl_libraries/my_CW/components/my_CW_mult/bindings
    ./ test1
Related Information

CWD Component in ChipWare Developer

Related commands:  
\texttt{hdI_create component} on page 188  
\texttt{hdI_create implementation} on page 190  
\texttt{hdI_create library} on page 192  
\texttt{hdI_create operator} on page 193  
\texttt{hdI_create package} on page 194  
\texttt{hdI_create parameter} on page 196  
\texttt{hdI_create pin} on page 198
**hdl_create component**

```plaintext
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:

  ```plaintext
  rc:/hdl_libraries/CW/components> hdl_create component CW_sweet_div
  rc:/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:

  ```plaintext
  rc:/> hdl_create component CW_sweet_div /hdl_libraries/CW/
  rc:/> ls /hdl_libraries/CW/components
  ...
  CW_sweet_div
  ...
Related Information

See the following sections in ChipWare Developer

- CWD Component
- ChipWare Registration

Related commands:  
- `hdl_create binding` on page 186
- `hdl_create implementation` on page 190
- `hdl_create library` on page 192
- `hdl_create operator` on page 193
- `hdl_create package` on page 194
- `hdl_create parameter` on page 196
- `hdl_create pin` on page 198
**hdl_create implementation**

```bash
hdl_create implementation implementation_name
    [-v1995 | -v2001 | -vhdl87 | -vhdl93] [-config string]
    [hdl_comp | implementations]
```

Creates an implementation for a ChipWare component. All implementations created with this command have a default priority of 1. A ChipWare implementation is also known as an architecture of the component. You must specify a language version.

**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.


**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:

  ```bash
  rc:/hdl_libraries/CW/components/CW_sweet_div> hdl_create implementation \   krystal -vhdl93
  rc:/hdl_libraries/CW/components/CW_sweet_div/implementations> ls
  ./ krystal
  ```

- 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:

  ```bash
  rc:/> hdl_create implementation krystal -vhdl93 \   /hdl_libraries/CW/components/CW_sweet_div
  rc:/> ls /hdl_libraries/CW/components/CW_Sweet_div/implementations/ \   krystal
  ```
Related Information

See the following sections in ChipWare Developer

- CWD Component
- Chipware Installation and Registration

Affects this attribute: priority

Related commands:
- hdl_create_binding on page 186
- hdl_create_component on page 188
- hdl_create_library on page 192
- hdl_create_operator on page 193
- hdl_create_package on page 194
- hdl_create_parameter on page 196
- hdl_create_pin on page 198
**hdl_create library**

**hdl_create library library_name**

Creates an 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 *ChipWare Developer*

- **CWD Component**
- **ChipWare Installation and Registration**

Related commands:

- `hdl_create binding` on page 186
- `hdl_create component` on page 188
- `hdl_create implementation` on page 190
- `hdl_create operator` on page 193
- `hdl_create package` on page 194
- `hdl_create parameter` on page 196
- `hdl_create pin` on page 198
**hdl_create operator**

```
hdh_create operator operator_name
    [-signed | -unsigned]
```

Creates a synthetic operator. The default operator type is unsigned.

**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. This is the default setting.

**Related Information**

**Synthetic Operator in ChipWare Developer**

Related commands:
- `hdl_create binding` on page 186
- `hdl_create component` on page 188
- `hdl_create implementation` on page 190
- `hdl_create library` on page 192
- `hdl_create package` on page 194
- `hdl_create parameter` on page 196
- `hdl_create pin` on page 198
**hdl_create package**

```plaintext
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 RTL Compiler 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:

  ```
  rc:/hdl_libraries/ieee/packages> hdl_create package numeric_std -path 
  /home/krystal/vhdl/packages/numeric_std.vhdl
  rc:/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:

  ```
  rc:/> hdl_create package numeric_std -path /home/krystal/vhdl/packages/numeric_std
  rc:/> ls /hdl_libraries/ieee/packages/
  ./ numeric_std
  ```
Related Information

Related commands:  
  hdl_create binding on page 186
  hdl_create component on page 188
  hdl_create implementation on page 190
  hdl_create library on page 192
  hdl_create operator on page 193
  hdl_create parameter on page 196
  hdl_create pin on page 198
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
ture. 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:

  rc:/hdl_libraries/CW/components/CW_sweet_div> hdl_create parameter WIDTH
  rc:/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:

  rc:/> hdl_create parameter WIDTH /hdl_libraries/CW/components/CW_sweet_div
  rc:/> ls /hdl_libraries/CW/components/CW_sweet_div/parameters/
  ./ WIDTH
Related Information

See the following sections in ChipWare Developer

- CWD Component
- ChipWare Installation and Registration

Affects this attribute: hdl_parameter
Related commands:
- hdl_create binding on page 186
- hdl_create component on page 188
- hdl_create implementation on page 190
- hdl_create library on page 192
- hdl_create operator on page 193
- hdl_create package on page 194
- hdl_create pin on page 198
hdl_create pin

```
hdle_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:

  ```
  rc:/hdl_libraries/CW/components/CW_sweet_div> hdl_create pin -input div_in
  rc:/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:

  ```
  rc:/> hdl_create pin -input div_in
  rc:/> ls /hdl_libraries/CW/components/CW_sweet_div/pins/
  ./   div_in
  ```
Related Information

See the following sections in ChipWare Developer

- CWD Component
- ChipWare Installation and Registration
- Synthetic Operator

Related commands:  
  - hdl_create binding on page 186
  - hdl_create component on page 188
  - hdl_create implementation on page 190
  - hdl_create library on page 192
  - hdl_create operator on page 193
  - hdl_create package on page 194
  - hdl_create parameter on page 196
Input and Output

- decrypt on page 204
- encrypt on page 205
- export_critical_endpoints on page 208
- read_db on page 210
- read_def on page 211
- read_dfm on page 212
- read_dft_abstract_model on page 214
- read_encounter on page 215
- read_hdl on page 216
- read_io_speclist on page 220
- read_memory_view on page 221
- read_netlist on page 222
- read_pmbist_interface_files on page 224
- read_power_intent on page 225
- read_saif on page 226
- read_sdc on page 227
- read_spef on page 229
- read_tcf on page 230
- read_vcd on page 231
- restore_design on page 232
- split_db on page 234
- write_atpg on page 235
write_bsd on page 236
write_compression_macro on page 237
write_db on page 238
write_def on page 240
write_design on page 241
write_dft_abstract_model on page 243
write_dft_rtl_model
write_do_ccd on page 245
write_do_ccd_cdc on page 246
write_do_ccd_compare_sdc on page 247
write_do_ccd_generate on page 249
write_do_ccd_validate on page 253
write_do_clp on page 254
write_do_lec on page 257
write_do_verify_cdc on page 261
write_encounter on page 263
write_et_atpg on page 266
write_et_bsv on page 267
write_et_lbist on page 269
write_et_lbist on page 269
write_et_mbist on page 270
write_et_no_tp_file on page 271
write_et_rrfa on page 272
write_ets on page 273
write_ett on page 274
write_forward_saif on page 275
write_hdl on page 276
write_io_speclist on page 280
■ **write_ldb** on page 281
■ **write_logic_bist_macro** on page 282
■ **write_mbist_testbench** on page 283
■ **write_pmbist_interface_files** on page 284
■ **write_pmbist_testbench** on page 285
■ **write_sai** on page 287
■ **write_scandef** on page 288
■ **write_script** on page 289
■ **write_sdc** on page 292
■ **write_sdc** on page 292
■ **write_sdf** on page 296
■ **write_set_load** on page 301
■ **write_spef** on page 302
■ **write_sv_wrapper** on page 303
■ **write_tcf** on page 306
■ **write_template** on page 307
decrypt

decrypt [-keydb path] file

Decrypts and evaluates a Tcl file that was encrypted with the encrypt command.

Options and Arguments

file
Specifications the name of the Tcl file to decrypted and evaluated.

-keydb path
Sets the NC PROTECT_KEYDB environment variable to the directory containing the public key needed to decrypt the file.

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.

rc:/> encrypt -tcl my_script.g > my_script_encr.g
rc:/> decrypt my_script_encr.g

Related Information

Related command: encrypt on page 205
encrypt

\texttt{encrypt \ inputfile\_name}
\begin{itemize}
  \item [-vlog | -vhdl | -tcl] [-pragma]
  \item [>] \ file
\end{itemize}

Uses the NC Protect protection scheme to encrypt the specified HDL or Tcl files.

\begin{itemize}
  \item Tip
  To load encrypted HDL files, use the \texttt{read\_hdl} command.
\end{itemize}

The command to source an encrypted Tcl file depends on the file extension of the encrypted Tcl file:
\begin{itemize}
  \item If the encrypted file does not have the \texttt{.etf} extension, use the \texttt{decrypt} command.
  \item If the encrypted file has the \texttt{.etf} extension, you can use the \texttt{source} command.
\end{itemize}

\section*{Options and Arguments}

\begin{itemize}
  \item \texttt{input\_file\_name} Specifies the file to be encrypted.
  \item \texttt{file} Specifies the name of the encrypted file.
    By default, the encrypted file is printed to standard out.
  \item [-pragma] Specifies to only encrypt the text between the \texttt{protect begin} and \texttt{protect end} NC Protect pragmas.
  \item [-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 \texttt{hidden\_proc}.
  \item [-vhdl] Uses VHDL style comments for NC Protect pragmas.
  \item [-vlog] Uses Verilog style comments for NC Protect pragmas. This is the default option.
\end{itemize}

\section*{Example}

- The following example encrypts the \texttt{ksable.vhdl} VHDL file, with VHDL constructs, to a file named \texttt{ksable\_encrypted.vhdl}. The encrypted file is then loaded.

  \begin{verbatim}
  rc:/> encrypt -vhdl ksable.vhdl > ksable\_encrypted.vhdl
  rc:/> read\_hdl -vhdl ksable\_encrypted.vhdl
  \end{verbatim}
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.

```verilog
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`.

```
rc:> 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.

```vhdl
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`.

```
rc:/> encrypt -vhdl -pragma org.vhdl > enc.vhdl
```

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.

```tcl
# 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
```
rc:/> encrypt -tcl test.tcl > test_enc.tcl
rc:/> info body im_hidden
rc:/> info body im_visible

# 'info' command will return data for this proc

Related Information

Related command: decrypt on page 204
export_critical_endpoints

export_critical_endpoints
    -rc_file string -fe_file string
    [-group | -no_group] [-verbose]
    [-percentage_of_endpoints integer]
    [-no_of_bins integer]
    [-percentage_difference integer] [-rtl]
    [-design string] [> file]

Generates a path adjust file, which allows synthesis to provide better timing closure results to Encounter.

Options and Arguments

-design string Specifies the module name.
-fe_file string Specifies the First Encounter (FE) 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 RTL Compiler endpoint report that you want to compare.
-rtl
  Writes a path adjust file that can be applied to the RTL.

-verbose
  Specifies a verbose report.

Related Information

*Path Adjust Flows* in *Encounter RTL Compiler Synthesis Flows*
**read_db**

```bash
read_db
  [db_file | -from_tcl string]
  [-quiet] [-verbose]
```

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.
- `from_tcl string` Specifies to read the specified Tcl object.
- `quiet` Suppresses any warning messages. Error messages are printed.
- `verbose` Enables verbose output while reading in the database file.

**Related Information**

**Using the RTL Compiler Database** in *Using Encounter RTL Compiler*

Related commands:

- `split_db` on page 234
- `write_db` on page 238
read_def

Refer to read_def in Chapter 10, “Physical.”
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, RTL Compiler 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
<?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> fflop1
    <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:

```bash
rc:/> read_dfm test1.dfm
rc:/> read_dfm test2.dfm
```
Related Information

Design For Manufacturing Flow in *Encounter RTL Compiler Synthesis Flows*

Affects these commands: report gates -yield
                           report yield

Affects this attribute: yield

Related attribute: optimize yield
read_dft_abstract_model

Refer to read_dft_abstract_model in Chapter 11, “Design for Test.”
read_encounter

Refer to read_encounter in Chapter 10, “Physical.”
**read_hdl**

```
read_hdl file_list
  [-v2001 | -v1995 | -sv | -vhdl ]
  [-library library_name[=library_name2]...]
  [-netlist]
  [-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 `rc -E -f <your script>` command to specify that RTL Compiler automatically quit if a script error is detected when reading in HDL files instead of holding at the rc> 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.

- `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 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.

-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.

See Reading a Partially Structural Design in Using Encounter RTL Compiler for detailed information on using the -netlist option to read and elaborate a partially structural design.

Note: If this option is specified, all the following options are ignored: -v1995,-v2001,-vhdl,-sv.

-sv

Specifies that the HDL files conform to SystemVerilog 3.1.a.

-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.
Examples

- The following example first loads the example1.v file, then the example2.v file:
  
  ```
  rc:/> 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
  ```

- 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:

```
rc:/> 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/
```

```
rc:/> ls /hdl_libraries/lib1/architectures/
```

```
rc:/> ls /hdl_libraries/lib2/architectures/
```

```
```
/hdl_libraries/lib2/architectures:
./compute/

Related Information

Reading a Partially Structural Design in Using Encounter RTL Compiler.

Affects this command: elaborate on page 360
Related command: read_netlist
Affects this attribute: hdl_parameter
Affected by these attributes: hdl_search_path
hdllanguage
hd_preserve_dangling_output_nets
hdilverilog_defines
read_io_speclist

Refer to read_io_speclist in Chapter 11, “Design for Test.”
read_memory_view

Refer to read_memory_view in Chapter 11, “Design for Test.”
read_netlist

read_netlist file_list
  [-top top_module_name]
  [-define macro=value]...
  [-v2001]

Reads and elaborates a Verilog 1995 structural netlist when the design does not include behavioral (VHDL or Verilog) modules.

A structural Verilog file contains only structural Verilog-1995 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 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.
-v2001 Specifies that the netlist files contains Verilog-2001 attributes. The tool can read in attributes with the following format:

`(* attribute[=value] [, attribute[=value] ...] *)...`

The opening and closing parentheses and the stars (*) must be entered literally in the Verilog files.

**Related Information**

Reading and Elaborating a Structural Netlist Design and Reading a Partially Structural Design in *Using Encounter RTL Compiler*.

**Related Commands:**
- `read_hdl -netlist`
- `write_hdl` on page 276

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`
read_pmbist_interface_files

Refer to read_pmbist_interface_files in Chapter 11, “Design for Test.”
read_power_intent

Refer to read_power_intent in Chapter 13, “Advanced Low Power Synthesis.”
read_saif

Refer to read_saif in Chapter 12, “Low Power Synthesis.”
read_sdc

read_sdc file
   [-stop_on_errors] [-no_compress]
   [-mode mode_name]

Reads a constraints file in Synopsys Design Constraint (SDC) format into RTL Compiler. RTL Compiler 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 RTL Compiler 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 RTL Compiler 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 RTL Compiler when ungrouping). The SDC file will have constraints reflecting these buffers.

Unsupported Constraints

Not all SDCs are supported. For those that are not supported, RTL Compiler will issue a warning message but store them for output for the write_sdc command. RTL Compiler 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
Options and Arguments

`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.

Related Information

**Applying Design Constraints** in *Using Encounter RTL Compiler*

**Performing Multi-Mode Timing Analysis** in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affects this command: `synthesize` on page 377
`create_mode` on page 315

Related command: `write_sdc` on page 292

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`
`timignore_data_check_for_non_endpoint_pins`
read_spef

Refer to read_spef in Chapter 10, “Physical.”
read_tcf

Refer to read_tcf in Chapter 12, “Low Power Synthesis.”
read_vcd

Refer to read_vcd in Chapter 12, “Low Power Synthesis.”
**restore_design**

`restore_design
  -db_dir string  -design_name design
  [-def file]   [-worst_corner string]
`

Loads the database written out by the Encounter® tool in the RTL Compiler tool.

In the absence of a configuration file in the Encounter database directory, the `restore_design` command reads the required design and library information from the `globals` and `viewDefinition` files in the Encounter database directory.

**Options and Arguments**

- **-db_dir string**
  Specifies the path to the Encounter database directory.

- **-def file**
  Specifies the path to the DEF file.

  If this option is not specified, the tool searches for a `design.def` or `design.def.gz` file in the Encounter database directory. If neither file is found, an error message is issued.

- **-design_name design**
  Specifies the name of the design.

  The design name is the base filename for the output files generated by Encounter tool.

- **-worst_corner string**
  Specifies the worst case delay corner of all corners defined in the `viewdefinitions.tcl` file.

  This option is required for multi-mode multi-corner designs.

  **Note:** The libraries and captable for the worst corner will be loaded in the RTL Compiler tool.

**Example**

The following command reads the Encounter database from the `fe_db_dat` directory, specifies that the name of the design is `test`. The assumptions are that a `test.def` or
test.def.gz file is part of the fe_db_dat directory and that the design is a non-MMMC design.

restore_design -db_dir fe_db_1.dat -design test

Related Information

Affects this command: 

synthesize on page 377
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.

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.

Related Information

Using the RTL Compiler Database in Using Encounter RTL Compiler

Related commands: read_db on page 210
                   write_db on page 238
write_atpg

Refer to write_atpg in Chapter 11, “Design for Test.”
write-bsdl

Refer to write-bsdl in Chapter 11, “Design for Test.”
write_compression_macro

Refer to write_compression_macro in Chapter 11, “Design for Test.”
write_db

write_db
   [-to_file db_file]
   [-all_root_attributes | -no_root_attributes]
   [-script file] [design] [-quiet] [-verbose]

Writes the netlist to a database file or returns a Tcl list. Root attributes with non-default settings can be included in the database or written to a script. By default, only root attributes affecting the QOR are written out.

Options and Arguments

-all_root_attributes
   Writes out all the root attributes with non-default settings.

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.
   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 db_file
   Specifies the name of the database file to be written.
   By default, the command returns a Tcl list.

-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

Using the RTL Compiler Database in Using Encounter RTL Compiler

Related commands:  
- `define_attribute` on page 1091
- `read_db` on page 210
- `split_db` on page 234
- `write_design` on page 241
write_def

See write_def on page 624 in Chapter 10, “Physical.”
write_design

write_design
    [-basename string] [-gzip_files] [-tcf]
    [-encounter] [-hierarchical] [design]

Generates all the files needed to reload the session in RTL Compiler (for example, .g, .v, and .tcl files). If you want to generate all the files that are need to loaded in both a RTL Compiler and Encounter® session, use the -encounter option.

**Note:** When you use this command on a design that is not fully mapped, for example after elaborate or synthesize -to_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.

When you write out the design after spatial optimization (synthesize -spatial), an encrypted file (basename.spl.etf) is written. To reload this file, use the decrypt command.

**Options and Arguments**

- **-basename string** Specifies the path and basename for the generated files.
- **design** Specifies the top-level design in RTL Compiler.
- **-encounter** Generates the additional files needed for Encounter.
- **-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.
- **-tcf** Specifies to write out a TCF containing the asserted switching activities of the pins in the design.
Example

The following example writes both the RTL Compiler and Encounter files as well as specifies the path and basename to be test/top:

rc:/> write_design -encounter -basename test/top
unix> ls /home/mydir/test
top.conf
top.g
top.rc_setup.tcl
top.v
top.enc_setup.tcl
top.mode
top.sdc
top.derate.tcl

Related Information

Generating Design and Session Information in Using Encounter RTL Compiler

Saving and Restoring a Session in RTL Compiler in Using Encounter RTL Compiler
write_dft_abstract_model

Refer to write_dft_abstract_model in Chapter 11, “Design for Test.”
write_dft_rtl_model

Refer to write_dft_rtl_model in Chapter 11, “Design for Test.”
write_do_ccd

write_do_ccd {cdc | compare_sdc | generate | propagate | validate}

Translates RTL Compiler 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**

Interfacing with Encounter Conformal Constraint Designer in *Interfacing Between RTL Compiler and Conformal*

Related commands:

- `write_do_ccd cdc` on page 246
- `write_do_ccd compare_sdc` on page 247
- `write_do_ccd generate` on page 249
- `write_do_ccd propagate` on page 251
- `write_do_ccd validate` on page 253
write_do_ccd cdc

write_do_ccd cdc
  [-design string] [-sdc files]
  [-no_exit] [-logfile file] [> file]

Writes a dofile for the Encounter® Conformal® Constraint Designer (CCD) for clock domain crossing checks.

Options and Arguments

- **-design string**  
  Specifies the top-level design in RTL Compiler.  
  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 thorough read_sdc.

Example

write_do_ccd cdc -sdc test.sdc -design test
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 Encounter® 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 RTL Compiler.
- **-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
Related Information

Comparing SDC Constraint Files in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

Related command: `compare_sdc` on page 410
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 Encounter® 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 RTL Compiler.

- **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.

- **-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

Using the Generate Flow with Dofiles in Interfacing Between Encounter RTL Compiler and Encounter Conformal

Affected by this attribute: wccd_threshold_percentage
**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 Encounter® 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 RTL Compiler.

- **-file string**
  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.
Example

The following command creates a do file to propagate the block-level SDC files i1.sdc and i2.sdc to the top level.

```
rc:/> 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
```
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 RTL Compiler.

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.

Related Information

Using the Validate Flow with Dofiles in Interfacing Between Encounter RTL Compiler and Encounter Conformal
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.


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 Encounter® 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 RTL Compiler.

- 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.
-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.
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/tcbn65lplvtwc10d90d72.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

Interfacing with Conformal Low Power in Interfacing Between Encounter RTL Compiler and Encounter Conformal

Affected by these attributes:  

`wclp_lib_statetable`
write_do_lec


Translates RTL Compiler settings to Encounter® Conformal® Logical Equivalence Checking commands.

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.

If you expect name changes in the CPF design objects, you must generate this file using the write_power_intent -cpf command.
-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.

-env_var string

Specifies the names and values of UNIX environment variables to be used in library, design, and log filenames in the dofile.

file

Redirects all the output to the specified file.

-flat

Performs a flattened Conformal LEC comparison.

-golden_design string

Specifies the UNIX path to an alternative golden design.

If the file was loaded into RTL Compiler (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.

-hier

Performs a hierarchical Conformal LEC comparison.

-logfile string

Specifies the name of the Conformal LEC logfile.

-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 RTL Compiler (before using the commit_power_intent command)

-pre_compare string

Specifies an extra dofile that must be sourced from the dofile before the LEC compare command.

-pre_exit string

Specifies an extra dofile that must be sourced from the dofile before the LEC exit command.
-**pre_read string**  Specifies an extra dofile that must be sourced from the dofile before the library and design are read.

-**revised_design string**  Specifies the UNIX path to the revised design.

-**save_session string**  Specifies the filename to save the LEC session.

-**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.

-**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 RTL Compiler.

-**verbose**  Generates a dofile with verbose reporting.

### Examples

- The following command will source the extra_settings.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 \
  -logfile cw_sim_sv.log > cw_sim.sv_lec.do

### Related Information

- [Interfacing with Encounter Conformal Logical Equivalence Checker](Interfacing Between Encounter RTL Compiler and Encounter Conformal) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

- [Performing a Low Power Equivalence Check](Performing a Low Power Equivalence Check) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

- [More About the write_do_lec dofile](More About the write_do_lec dofile) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

Related command:  [write_power_intent](Related command: write_power_intent) on page 1038
Affected by these attributes:

- boundary_optimize_invert_hier_pins
- wlec_add_noback_box_retime_subdesign
- wlec_analyze_abort
- wlec_analyze_setup
- wlec_auto_analyze
- wlec_compare_threads
- wlec_cut_point
- wlec_hier_comp_threshold
- wlec_lib_statetable
- wlec_set_cdn_synth_root
- wlec_uniquify
- wlec_use_lec_model
write_do_verify cdc

write_do_verify cdc [-categorize | -validate]
   -sdc string
   [ -design string] [-no_exit]
   [-logfile string] [> file]

Generates a dofile for Encounter® 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. RTL Compiler 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 RTL Compiler.
- `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
  ```
Related Information

Interfacing with Encounter Conformal Extended Checks in Interfacing Between Encounter RTL Compiler and Encounter Conformal

Affected by these attributes:

- wcdc_clock_dom_comb_propagation
- wcdc_synchronizer_type
write_encounter


Writes Encounter® input files to a single directory. The command will only convert library domains into power domains for Encounter if power domains exist in RTL Compiler. 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 Encounter.

The generated files are all required Encounter input files and include the following files:

- Netlist (.v)
- Encounter global variables file (.globals) or Encounter 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 Encounter data in an Encounter session. This eliminates the need to load each of the necessary files sequentially.

The .mode file contains all the Encounter 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 synthesize -to_placed. However, RTL Compiler 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.

Note: The MSV (multiple supply voltage) library domain setup commands require Encounter version 4.2 or later. Multiple timing mode is supported in Encounter version 5.2 or later.
Options and Arguments

-basename string  Specifies the directory path name and base filename for the output data. The default directory is ./rc_enc_des and the default filename without the extension is rc.

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—Encounter 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 Encounter Timing System) 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 RTL Compiler 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 Encounter configuration file to use as a template for the generated configuration file.

  **Note:** The configuration file is used in the Legacy Configuration Data Flow.
-referenceGlobalsFile globals_file

Specifies a reference Encounter globals file to use as a template for the generated global variables file.

Use this option with the init_design simple data flow in the EDI System.

-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 Encounter input files for the test07 design to the directory TEST. The basename for the files are specified to be test1.

```
rc:/> 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:

```
rc:/> 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

Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Export to Place and Route in The Multiple Supply Voltage Flow in Low Power in Encounter RTL Compiler.

Related commands:
create_mode on page 315
read_encounter on page 215
write_et_atpg

Refer to write_et_atpg in Chapter 11, “Design for Test.”
write_et_bsv

Refer to write_et_bsv in Chapter 11, “Design for Test.”
write_et_dfa

Refer to write_et_dfa in Chapter 11, “Design for Test.”
write_et_lbist

Refer to `write_et_lbist` in Chapter 11, “Design for Test.”
write_et_mbist

Refer to write_et_mbist in Chapter 11, “Design for Test.”
write_et_no_tp_file

Refer to write_et_no_tp_file in Chapter 11, “Design for Test.”
write_et_rrfa

Refer to write_et_rrfa in Chapter 11, “Design for Test.”
write_ets

write_ets [-default] [-ocv]
   [-pre_include string] [-post_include string]
   [-netlist string]
   [-sdc string] [-sdf string] [-spef string]
   [> file]

Generates an 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 file 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 file generated by write_ets.

-**sdc string**
  Specifies the UNIX path of the SDC file.

-**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.
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.
write_forward_saif

Refer to write_forward_saif in Chapter 12, “Low Power Synthesis.”
write_hdl

write_hdl  {design|subdesign}...
    [-suffix string]
    [-abstract] [-generic] [-depth integer]
    [-equation] [-lec] [-v2001] [ > 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.
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 command has been run, you cannot recover the version of the design that was generated using this option prior to synthesis.

-lec
Generates an intermediate netlist with additional information to facilitate formal verification with Encounter® Conformal® Equivalence Checking. See Related Information for more information on when to write out the intermediate netlist.

-suffix
Specifies the string to be appended to the name of all defined modules in the generated netlist.

-v2001
Writes out the Verilog-2001 attributes stored with instances, ports and subports in the design.

Examples

- The following example writes out the logic abstract model definition of design test:
  rc:/> write_hdl -abstract

  // Generated by Cadence RTL Compiler-D (RC) version

  module test(in1, in2, out1, out2, clk1, clk2, clk3, sel, se2);
  input [3:0] in1;
  input [7:0] in2;
  input clk1, clk2, clk3, sel, 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:
  rc:/> 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:
  rc:/> set_attr unresolved true [get_attr instance [get_attr subdesign bottom]]
  rc:/> 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):
  rc:/> read_hdl design.v
  rc:/> ...
  rc:/> synthesize -to_mapped
  rc:/> ...
  rc:/> write_hdl
The following example replaces each Verilog primitive gate by an equivalent Verilog assign statement:

```
rc:/> write_hdl -equation design.v
```

The following example writes out the design as generic logic regardless of its current mapped state:

```
rc:/> write_hdl -generic > design_rtl.v
```

Related Information

Writing Out the Design Netlist in *Using Encounter RTL Compiler*.

Creating a Logic Abstract Model in Design for Test in Encounter RTL Compiler

When to Write out Intermediate Netlist in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

Affected by these commands: elaborate on page 360
synthesize on page 377

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
- 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
write_io_speclist

Refer to write_io_speclist in Chapter 11, “Design for Test.”
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.
write_logic_bist_macro

Refer to write_logic_bist_macro in Chapter 11, “Design for Test.”
write_mbist_testbench

Refer to write_mbist_testbench in Chapter 11, “Design for Test.”
write_pmbist_interface_files

Refer to write_pmbist_interface_files in Chapter 11, “Design for Test.”
write_pmbist_testbench

Refer to write_pmbist_testbench in Chapter 11, “Design for Test.”
write_power_intent

Refer to write_power_intent in Chapter 13, “Advanced Low Power Synthesis.”
write_saif

Refer to write_saif in Chapter 12, “Low Power Synthesis.”
write_scandef

Refer to write_scandef in Chapter 11, “Design for Test.”
write_script

write_script [ -hd1 ]
 [ -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 RTL Compiler 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.
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 RTL Compiler session, the RC-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:
  ```
  rc:/> write_hdl > mapped.v
  rc:/> write_script > mapped.g
  ```

  The design and script is subsequently read into another RTL Compiler 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.

  ```
  rc:/> set_attribute library areid.lib
  rc:/> set_attribute lef_library areid.lef
  rc:/> set_attribute cap_table_file areid.cap
  rc:/> read mapped.v
  rc:/> elaborate
  rc:/> source mapped.g
  ```

- The following example automatically compresses the output file using the .gz extension:
  ```
  rc:/> write_script > foo.g.gz
  ```
Related Information

Affected by these commands:

- `create_mode` on page 315
- `define_clock` on page 318
- `define_cost_group` on page 323
- `define_dft_scan_chain` on page 739
- `external_delay` on page 326
- `multi_cycle` on page 332
- `path_adjust` on page 336
- `path_delay` on page 340
- `path_disable` on page 343
- `path_group` on page 346
write_sdc

write_sdc
    [-version {1.1|1.3|1.4|1.5|1.5rc|1.7}]
    [-strict] [-no_split] [-exclude cmd_list]
    [-mode mode_name] [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, RTL Compiler 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: rc:/> write_sdc -version 1.1.

For those SDCs that are not supported, RTL Compiler will issue a warning message but store them for output for the write_sdc command. RTL Compiler 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

design
    Specifies the name of the design for which to write the SDC constraints.

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.
Examples

- The following example writes out the SDC constraints to the my_des.sdc file:
  rc:/> 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]~

- 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 example shows how to write out mode-specific constraints:
  write_sdc -mode model1 model1.sdc

- 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:

-version {1.1|1.3|1.4|1.5|1.5rc|1.7}

  Specifies the SDC version to use. Version 1.5rc includes the set_time_unit and set_load_unit commands.

  Default: 1.7
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.

rc:/> write_sdc
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # Created by Encounter(R) RTL Compiler 10.1.200 on Tue Nov 16 12:58:56 -0800 2010
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
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 abc]

rc:/> write_sdc -exclude "set_clock_latency set_clock_uncertainty \\nset_case_analysis create_clock"

or

rc:/> write_sdc -exclude {set_clock_latency set_clock_uncertainty \\nset_case_analysis create_clock}

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # Created by Encounter(R) RTL Compiler 10.1.200 on Tue Nov 16 13:05:12 -0800 2010
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
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

Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Affected by this command: 
create_mode on page 315
read_sdc on page 227
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}] [-condeIs]
  [-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.

**Note:** Use the write_sdf command after technology mapping (after the `synthesize-to_mapped` 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—Excludes cell delays and timing checks from being written into the SDF file.
  
  nochecks—Only excludes the timing checks.
  
  **Default:** all

- **condelse**
  
  Writes CONDELSE constructs with the default value when a COND construct is written.

- **delimiter character**
  
  Specifies the hierarchy divider character to be used in the SDF file. The valid options are the “/” and “.” characters.

- **design**
  
  Specifies the design name for which the SDF file has to be generated.
-edges \{edged \mid check\_edge\}

Specifies the edges values.

\texttt{check\_edge}—Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs.

\texttt{edged}—Keeps edge specifiers on timing check arcs as well as combinational arcs.

\textit{Default:} edged

\texttt{file}

Specify the SDF file name.

-\texttt{-interconn} \{port \mid interconnect\}

Specifies the construct to use for writing out net delays.

\texttt{port}—Writes out the net delays using the \texttt{PORT} construct.

\texttt{interconnect}—Writes out the net delays using the \texttt{INTERCONNECT} construct.

\textit{Default:} port

-\texttt{-no\_escape}

Writes out object names without escaping the special characters such as \texttt{“[“ or “]”}.

-\texttt{-nongeckehces}

Converts all negative timing check values to 0.0.

-\texttt{-no\_empty\_cells}

Suppresses writing out empty cell descriptions.

-\texttt{-no\_input\_port\_nets}

Suppresses writing out nets connected to input ports.

-\texttt{-no\_output\_port\_nets}

Suppresses writing out nets connected to output ports.

-\texttt{-nospitf\_tuning\_check}

Does not split the \texttt{TIMINGCHECK} delays (SETUP/HOLD/RECOVERY/REMOVAL delays). Instead, the maximum delay values are used.

-\texttt{-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

-setushold [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:

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 \texttt{SETUP} and a \texttt{HOLD} check exist, the tool writes a combined \texttt{SETUPHOLD} check in the SDF file as follows:

\begin{verbatim}
SETUPHOLD (value) (value) (value) (value)
\end{verbatim}

However, if only the \texttt{SETUP} check exists, the tool only writes the \texttt{SETUP} check in the SDF file:

\begin{verbatim}
SETUP (value) (value)
\end{verbatim}

\textit{Default: merge\_always}

\texttt{-timescale \{ps | ns\}}

Specifies the timescale setting of the SDF file in either nanoseconds or picoseconds.

\textit{Default: ps}

\texttt{-version \{OVI 3.0 | OVI 2.1\}}

Specifies whether to generate SDF version 2.1 or 3.0.

\textit{Default: OVI 3.0}

\section*{Examples}

\textbullet\hspace{1em} The following example writes out the SDF file, \texttt{areid.sdf}, with the "/" delimiting character:

\begin{verbatim}
rc:/> synthesize -to\_mapped
rc:/> write\_sdf -delimiter "/" > areid.sdf
\end{verbatim}

\textbullet\hspace{1em} The following report illustrates two \texttt{TIMINGCHECK} examples: the first only writes out the maximum delays while the second writes out all the delays.

\begin{verbatim}
(TIMINGCHECK
 (HOLD D (posedge CK) (::0.0))
 (SETUP D (posedge CK) (::0.452))
)

(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))
)
\end{verbatim}
To write out only the maximum TIMINGCHECK delays (the first report above), use the
-nosplit_timing_check option:

rc:/> write_sdf -nosplit_timing_check > areid.sdf
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:

```
rc:/> 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]]
```
write_spef

See write_spef in Chapter 10, “Physical.”
write_sv_wrapper

write_sv_wrapper
  [-allow_error]
  [-module_suffix string] [-subdesign_suffix string]
  [-exclude_type_definition]
  [-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 RTL Compiler 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.

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.

- `-exclude_type_definition` Prevents to write out complex types (structs, unions, interfaces, and so on) for the wrapper module.

file Specifies the file to which to write the wrapper SystemVerilog module.

- `-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.
Example

Consider the following Verilog RTL input in test.v.

```verilog
// 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:

```bash
## test.g
set_attr library tutorial.lib
read_hdl -sv test.v
elaborate
synthesize -to_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.

```verilog
// 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:

```bash
## test.g
set_attr library tutorial.lib
read_hdl -sv test.v
set_attr hdl_sv_module_wrapper true /
elaborate
synthesize -to_map
write_sv_wrapper -update_design_name -subdesign_suffix _core test > test_wrap.v
write_hdl > test.vc
```
This results in the following output:

```vhdl
// 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

// 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 (struct, interface, union, enum, and so on) that occur in the portlist of the design, and not in its subdesigns.
write_tcf

Refer to write_tcf in Chapter 12, “Low Power Synthesis.”
write_template

write_template
   [-dft] [-power] [-cpf] [-retime]
   [-physical] [-performance]
   [-area] [-no_sdc] [-n2n] [-yield]
   [-full] [-simple] [-split]
   [-multimode] -outfile file

Generates a template script with the commands and attributes needed to run RTL Compiler. 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 RTL Compiler. 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 RTL Compiler 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.
Examples

- The following example writes out the basic template file with the constraints in SDC format:
  
  ```
  rc:// write_template -outfile template.g
  ```

- The following example writes out the basic template file with the constraints in the RTL Compiler format using the `define_clock` and the `external_delay` commands:
  
  ```
  rc:// 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:
  
  ```
  rc:// 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:
  
  ```
  rc:// 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:
  
  ```
  rc:// write_template -split -outfile template.g
  ```
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
```
Constraints

- clock_uncertainty on page 312
- create_mode on page 315
- define_clock on page 318
- define_cost_group on page 323
- derive_environment on page 324
- external_delay on page 326
- generate_constraints on page 330
- multi_cycle on page 332
- path_adjust on page 336
- path_delay on page 340
- path_disable on page 343
- path_group on page 346
- propagate_constraints
- specify_paths on page 351
- validate_constraints on page 357
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.
Command Reference for Encounter RTL Compiler

Constraints

```
-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]
  ```
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
Related command: dc::set_clock_uncertainty
Sets these attributes: clock_hold_uncertainty
                           clock_setup_uncertainty
                           hold uncertainty by clock
                           setup uncertainty by clock
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_name/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:

   rc:/>create_mode -name "model1 model2" -design design_name

- The following example creates two modes using multiple create_mode commands, and
  declares one of them as the default mode.

   rc:/> create_mode -name a
   /designs/design_name/modes/a
   rc:/>create_mode -name b -default
   /designs/design_name/modes/b
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

**Creating Modes** in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler* for detailed information.

Affects these commands:

- `define_clock` on page 318
- `external_delay` on page 326
- `multi_cycle` on page 332
- `path_adjust` on page 336
- `path_delay` on page 340
- `path_disable` on page 343
- `path_group` on page 346
- `read_sdc` on page 227
- `report clocks` on page 438
- `report timing` on page 541
- `specify_paths` on page 351
- `write_sdc` on page 292

Related commands:

- `derive_environment` on page 324
- `report summary` on page 535
- `write_encounter` on page 263
- `write_script` on page 289

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

(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
**define_clock**

```plaintext
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
-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. RTL Compiler 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.
Examples

- The following example defines a clock for a design with top module `alu`.
  ```shell
  rc:/> 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:
  ```shell
  rc:/> define_clock -period 10000 -name 300MHz -divide_period 3 [clock_ports]
  ```
  The clock period is 10,000/3 picoseconds. This allows RTL Compiler 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, RTL Compiler 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:
  ```shell
  rc:/> define_clock -period 10000 -name 100MHz -rise 20 -fall 80
  ```

- The following example defines clock 100MHz with the falling edge 1/3 and the rising edge 2/3 of the way through the period:
  ```shell
  rc:/> define_clock -period 10000 -name 100MHz -rise 2 -divide_rise 3 \n  ==> -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 RTL Compiler needs this precise definition to compute the correct timing constraints for paths that are launched by one clock and captured by another.
The following examples create a clock domain `system` and assign the original 100 MHz and 300 MHz clocks to it:

```
rc:/> define_clock -domain "system" -period 10000 -name 100MHz
rc:/> 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`:

```
rc:/> 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`:

```
rc:/> 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:

```
rc:/> find / -clock 100MHz
```

The following example examines the attributes of a clock object:

```
rc:/> ls -a [find / -clock 100MHz]
```

**Note:** The clock object is identified by its directory path.

**Related Information**

*Defining the Clock Period* in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.*

Affects these commands: `clock_ports` on page 409

`external_delay` on page 326

`multi_cycle` on page 332

`path_adjust` on page 336

`path_delay` on page 340

`path_disable` on page 343

`path_group` on page 346

`report clocks` on page 438

`report qor` on page 523
specify_paths on page 351
read_sdc on page 227
report_clocks on page 438
report_summary on page 535
report_timing on page 541
synthesize on page 377
write_encounter on page 263
write_script on page 289
write_sdc on page 292

Related command: create_mode on page 315

Affects these attributes: divide_fall
divide_period
divide_rise
fall
period
rise
propagated_clocks_by_mode
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

-name string
    Specifies the name of the cost group.
    Default: grp_x

-design design
    Specifies the name of the design for which you want to define the cost group.

-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.

rc:/> define_cost_group -name I2O -weight 1
/designs/.../timing/cost_groups/I2O

Related Information

Creating Path Groups and Cost Groups in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.

Generating Timing Reports in Using Encounter RTL Compiler

Affects these commands: path_group on page 346
                        report timing on page 541
                        write_script on page 289

Sets this attribute: weight
**derive_environment**

**derive_environment**

\[-\text{name} \] \[-\text{sdc\_only} \] \[-\text{power\_intent} \] \text{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 RTL Compiler’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 RTL Compiler 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 constraints, 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.
- `-sdc_only` Specifies that the tool only apply constraints to the new design that can be expressed in SDC.

**Important**

This option is only available as a limited access feature.

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.
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 RTL Compiler’s more powerful constraint language that produces a more accurate timing view, but is not understood by other tools.

**Related Information**

- **Synthesizing Submodules** in *Using Encounter RTL Compiler*
- **Writing out the Multi-Mode Environment** in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*
- **CPF Partitioning Flow** in *Low Power in Encounter RTL Compiler*

**Affected by this command:**
- `path_adjust` on page 336
- `create_mode` on page 315

**Affects these commands:**
- `report timing` on page 541
- `synthesize` on page 377

**Sets this attribute:**
- `precluded_path_adjusts`
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.
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:** RTL Compiler 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:** RTL Compiler does not support hold analysis.
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.

{\textit{pin} | \textit{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 \textit{a} relative to the falling edge of clock \textit{clock1}:

  ```
  rc:/> external_delay -input 300 -edge_fall -clock [find / -clock clock1] \ 
[find / -port a*]
  ```

  RTL Compiler 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 \textit{a} relative to the rising edge (default) of clock \textit{clock1}:

  ```
  rc:/> external_delay -output 1300 -clock [find / -clock clock1] \ 
[find / -port a*]
  ```

  RTL Compiler interprets this as a minimum setup time for external logic.

### Related Information

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

- Setting Input Delays
- Setting Output Delays
- Performing Multi-Mode Timing Analysis

Affected by this command: \textit{define_clock} on page 318
Affects these commands:  
- report qor on page 523  
- report timing on page 541  
- synthesize on page 377  
- write_encounter on page 263  
- write_sdc on page 292  
- write_script on page 289

Related commands:  
- create_mode on page 315  
- define_clock on page 318  
- derive_environment on page 324  
- multi_cycle on page 332  
- path_adjust on page 336  
- path_delay on page 340  
- path_disable on page 343  
- path_group on page 346  
- specify_paths on page 351

Affects these attributes:  
- External Delay Attributes

Related attributes:  
- (pin/port) external_delays_by_mode
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, RTL Compiler will internally generate the SDC constraints and verify them against the design at its current state. RTL Compiler 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 RTL Compiler 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.
-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:

```bash
rc:/> 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:

```bash
rc:/> generate_constraints -dfpgen -netlist generic.nl.v -in_sdc top.sdc -report top_out.rep -slack 2 -out_sdc top_out.sdc
```

Related Information

Using the Generate Flow without Dofiles in Interfacing between Encounter RTL Compiler and Encounter Conformal

Affected by this attribute:  
  wccd_threshold_percentage

Related command:  
  validate_constraints on page 357
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.

RTL Compiler 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.
   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 a 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.

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:
  
  rc:/> multi_cycle -through a -to b

- The following example requires a path to pass through either object a or b.
  
  rc:/> 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:
  
  rc:/> multi_cycle -through a -through b -through c

- The following example uses a Tcl variable to save the timing exception for future reference:
  
  rc:/> 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:
  
  rc:/> rm $two_cycle

- The following example searches for a timing exception that you defined earlier:
  
  rc:/> multi_cycle -capture_shift 2 -from [find / -port a] -name two_cycle
  
  /designs/alu/timing/exceptions/multi_cycles/two_cycle
  
  ...
  
  rc:/> 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:
  
  rc:/> \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

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

- Setting Timing Exceptions
- Overriding the Default Timing Constraint for Multi-Cycle Paths
- Performing Multi-Mode Timing Analysis

Affects these commands:

- `report qor` on page 523
- `report timing` on page 541
- `specify_paths` on page 351
- `synthesize` on page 377
- `write_encounter` on page 263
- `write_sdc` on page 292
- `write_script` on page 289

Related commands:

- `create_mode` on page 315
- `define_clock` on page 318
- `external_delay` on page 326
- `path_adjust` on page 336
- `path_delay` on page 340
- `path_disable` on page 343
- `path_group` on page 346
- `specify_paths` on page 351

Sets these attributes:

- Exception Attributes
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 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/../timing/exceptions/path_adjusts

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 a 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.

   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 removes the timing exception that you saved in the `override` variable:

```sh
rc:/> set override [path_adjust -to $clock -delay -500]
rc:/> rm $override
```

■ The following example searches for a timing exception that you defined earlier:

```sh
rc:/> path_adjust -to $clock -delay -500 -name override /designs/alu/timing/exceptions/path_adjusts/override
...  
rc:/> find / -exception override /designs/alu/timing/exceptions/path_adjusts/override
```

■ The following examples are equivalent and apply to paths starting from (clock) pin `clk1`:

```sh
path_adjust -from clk1
path_adjust -paths [specify_paths -from clk1]
```

Related Information

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

■ Creating Path Groups and Cost Groups

■ Modifying Path Constraints

■ Generating Post-Synthesis Timing Reports

■ Performing Multi-Mode Timing Analysis

Affects these commands:  

- `report timing` on page 541
- `report qor` on page 523
- `specify_paths` on page 351
- `synthesize` on page 377
- `write_script` on page 289
- `write_encounter` on page 263
- `write_sdc` on page 292
- `write_script` on page 289

Related commands  

- `define_clock` on page 318
- `external_delay` on page 326
multi_cycle on page 332
path_delay on page 340
path_disable on page 343
path_group on page 346
specify_paths on page 351

Sets these attributes: Exception Attributes
path_delay

path_delay -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]

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.

RTL Compiler 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.
-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.
  rc:/ 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.
  rc:/ path_delay -delay 4000 -to [find / -port b] -name my_delay /designs/alu/timing/exceptions/path_delays/my_delay
The following command searches for the timing exception my_delay defined earlier:

```
rc:/> 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**

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*:

- Creating Clock Domains
- Setting Timing Exceptions
- Modifying Path Constraints
- Performing Multi-Mode Timing Analysis

Affects these commands:  
- `report qor` on page 523
- `report timing` on page 541
- `specify_paths` on page 351
- `synthesize` on page 377
- `write_script` on page 289
- `write_sdc` on page 292

Related commands:  
- `create_mode` on page 315
- `define_clock` on page 318
- `external_delay` on page 326
- `multi_cycle` on page 332
- `path_adjust` on page 336
- `path_disable` on page 343
- `path_group` on page 346
- `specify_paths` on page 351

Sets these attributes:  
- Exception Attributes
path_disable

path_disable
{ {-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] [-name \string]}

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.

RTL Compiler 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.

  **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:
  rc:/> set false_path [path_disable -from [find / -port a]]

- The following example removes the timing exception that you saved in variable false_path:
  rc:/> rm $false_path

- The following example lists timing exception objects using the ls command.
  rc:/> ls -l timing/exceptions/path_disables
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**

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*:

- Specifying a False Path
- Performing Multi-Mode Timing Analysis

### Affects these commands:

- `report qor` on page 523
- `report timing` on page 541
- `specify_paths` on page 351
- `synthesize` on page 377
- `write_encounter` on page 263
- `write_script` on page 289
- `write_sdc` on page 292

### Affected by these commands:

- `define_clock` on page 318
- `multi_cycle` on page 332

### Related commands:

- `create_mode` on page 315
- `define_clock` on page 318
- `external_delay` on page 326
- `path_adjust` on page 336
- `pathDelay` on page 340
- `path_group` on page 346
- `specify_paths` on page 351

### Sets these attributes:

- Exception Attributes
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.
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

Creating Path Groups and Cost Groups in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler.

Performing Multi-Mode Timing Analysis in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Affects these commands:

- `define_cost_group` on page 323
- `report qor` on page 523
- `report timing` on page 541
- `specify_paths` on page 351
- `synthesize` on page 377
- `write_encounter` on page 263
- `write_script` on page 289
- `write_sdc` on page 292

Related commands:

- `create_mode` on page 315
- `define_clock` on page 318
external_delay on page 326
multi_cycle on page 332
path_adjust on page 336
path_delay on page 340
path_disable on page 343
specify_paths on page 351

Sets these attributes: Exception Attributes
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 350.

- **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**
  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 Encounter® 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 Encounter® 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.

  rc:/> 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 "\{\$WORKINGDIR/block1.sdc\} \n  \{\$WORKINGDIR/block2.sdc\} ..."

  or

  propagate_constraints -block_sdc [list \n      [\$WORKINGDIR/block1.sdc] \n      [\$WORKINGDIR/block2.sdc] ] ...
specify_paths

specify_paths [-mode mode_name] [-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

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}

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.
-through_fall_pin \{pin \mid port \mid 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 \mid port \mid 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 \mid port \mid clock \mid external_delay \mid 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 \mid port \mid clock \mid external_delay \mid 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 \mid port \mid clock \mid external_delay \mid 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.
-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.

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:
  rc:/> 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:
  rc:/> path_group -paths [specify_paths -from_rise_pin clk1] -group I20

- The following example uses the -to_fall_clock option with a pin object:
  rc:/> 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:
  rc:/> path_disable -paths [specify_paths -capture_clock_pins RAM/CK1]
Related Information

See the following sections in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler:

- Specifying Paths for Exceptions
- Performing Multi-Mode Timing Analysis

Affects these commands:
- `multi_cycle` on page 332
- `path_adjust` on page 336
- `path_delay` on page 340
- `path_disable` on page 343
- `path_group` on page 346
- `report qor` on page 523
- `report timing` on page 541
- `write_encounter` on page 263
- `write_script` on page 289
- `write_sdc` on page 292

Related commands:
- `create_mode` on page 315
- `define_clock` on page 318
- `external_delay` on page 326

Affects this attribute:
- `paths`
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, RTL Compiler 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:
  rc:/> validate_constraints -rtl -sdc /home/test/top.sdc

- The following command validates the generic.nl.v netlist against the lane.sdc:
  rc:/> validate_constraints -netlist /home/cody/design/netlist.nl.v \
   -sdc /home/cody/lane.sdc
Related Information

Using the Validate Flow without Dofiles in Interfacing between Encounter RTL Compiler and Encounter Conformal

Related command: generate constraints on page 330
Elaboration and Synthesis

- **elaborate** on page 360
- **get_remove_assign_options** on page 363
- **merge_to_multibit_cells** on page 364
- **remap_to_dedicated_clock_library** on page 365
- **remove_assigns_without_optimization** on page 366
- **remove_clock_reconvergence** on page 369
- **remove_inserted_sync_enable_logic** on page 370
- **retime** on page 371
- **set_remove_assign_options** on page 373
- **synthesize** on page 377
elaborate

elaborate [-parameters string] [module]...
    [-libpath path]... [-libext extension]...

Creates a design from a Verilog module or from a VHDL entity and architecture. Undefined modules and VHDL entities are labeled “unresolved” and treated as blackboxes.

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.

  If you omit module, all top-level modules 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.

RTL Compiler 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:

```
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 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
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 VHDL configuration to elaborate.

elaborate $config_name

Related Information

Affected by this command: read_hdl on page 216

Affected by this attribute: library
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 366
   set_remove_assign_options on page 373
merge_to_multibit_cells

merge_to_multibit_cells [-logical] design

Performs standalone mapping of single bit cells to multibit cells without performing any other optimizations.

Note: This command can only be run on a mapped design.

Options and Arguments

design

Specifies the name of the design for which to perform multibit mapping.

-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.

Related Information

Mapping to MultiBit Cells in Encounter RTL Compiler Synthesis Flows

Affected by these attributes:

use_multibit_cells
use_multibit_combo_cells
use_multibit_seq_and_tristate_cells
remap_to_dedicated_clock_library

```bash
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 RTL Compiler, clock networks are considered ideal.

**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).

  **Default:** medium
remove_assigns_without_optimization

remove_assigns_without_optimization
[-buffer_or_inverter <libcell>]
[-no_buffers_use_inverters]
[-allow_unloadedBuffers]
[-use_inverted_signal]
[-ignore_preserve_map_size_ok]
[-ignore_preserve_setting]
[-include_local_constant_assigns]
[-dont_respect_boundary_optimization]
[-preserve_dangling_nets]
[-dont_skip_unconstrained_optimization]
[-skip_hierarchical_subdesigns]
[-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.

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

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, RTL Compiler 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 RTL Compiler 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, RTL Compiler 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.

-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.

-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.

Related Information

Affected by this attribute: remove_assigns
remove_clock_reconvergence

remove_clock_reconvergence
   [-clocks clock...] [-effort {low|high}]
   [-ports port_list] [design]...

Optimizes the clock path by removing clock path reconvergence 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]"
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, RTL Compiler 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 RTL Compiler must write the list of flops from which the synchronous enable pins were removed.

Related Information

Related command: synchronize on page 377
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. If no option is specified, the -min_delay option is implied. Optimization is realized through appropriately moving registers. The area optimization will not be done at the expense of timing. That is, optimizing the area will not degrade the timing.

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 — RTL Compiler consumes as much time as necessary to provide the optimum timing solution.
- 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:

```

cr:/> retime -prepare
rc:/> retime -min_delay
```

**Related Information**

Retiming the Design in *Using Encounter RTL Compiler*

Related command:  
synthesize on page 377

Affected by these attributes:  
dont_retime

ret ime

retime_async_reset

retime_effort_level

retime_hard_region

retime_move_mux_loop_with_reg

retime_optimize_reset

re timing_clocks

Related attributes:  
re time_original_registers

re time_reg_naming_suffix

re time_verification_flow

trace_retime
set_remove_assign_options

set_remove_assign_options [-design {design | subdesign}]
  [ [-buffer_or_inverter libcell]]
  [-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

-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, RTL Compiler 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.

```bash
-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.

```bash
-dont_respect_boundary_optimization
```

Allows RTL Compiler to disconnect the unloaded subports of a subdesign, if the `boundary_opto` attribute on the subdesign was set to `false`.

```bash
-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, RTL Compiler skips unconstrained paths.

**Note:** This option has no effect in the RC-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 RC-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. RTL Compiler handles assign statements natively and does not require assign removal.

**Note:** To remove unconstrained paths in the RC-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.
Examples

- The following command specifies to use buffer BUFX2 to replace assign statements in subdesign med.

  ```
  rc:/> 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.

  ```
  rc:/> set_remove_assign_options -buffer BUFX2 [find / -subdesign bottom]
  rc:/> 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

For use with the CPF flow, refer to:

- Using CPF for Multiple Supply Voltage Designs in Low Power in Encounter RTL Compiler
- Using CPF for Designs Using Power Shutoff Methodology in Low Power in Encounter RTL Compiler
- Using CPF for Designs Using Dynamic Voltage Frequency Scaling in Low Power in Encounter RTL Compiler

Affects this command: `synthesize` -incremental

AFFECTED BY THIS ATTRIBUTE: `remove_assigns`
synthesize

synthesize [-effort {medium | low | high | express}]
   [-to_clock_gated] [-to_generic]
   [-to_mapped] [-to_placed | -spatial]
   [-auto_identify_shift_register]
       [-shift_register_min_length integer]
       [-shift_register_max_length integer]]
   [-incremental | -no_incremental] [design]...

Determines the most suitable design implementation using the given design constraints
(clock cycle, input delays, output delays, technology library, and so on).

The `synthesize` command takes a list of top-level designs, synthesizes the RTL blocks,
optimizes the logic, and performs technology mapping.

Options and Arguments

- `auto_identify_shift_register`
  Automatically identifies functional shift register segments.

- `design`
  Specifies the name of the design to synthesize.

  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.

- `effort {medium | low | high | express}`
  Specifies the effort to use during optimization.

  Default: medium

  The `express` effort enables the Express flow for both logical
  (-to_mapped) and physical (-to_placed) synthesis. The
  Express flow enables early feasibility analysis with much faster
  runtimes and reasonable quality of results.

  **Important**

  The `express` effort is only available as a limited
  access feature.
-incremental

Incrementally optimizes mapped gates. Allows the mapper to preserve the current implementation of the design and perform incremental optimizations if and only if the procedure guarantees an improvement in the overall cost of the design.

Note: This option only works with an already mapped design.

-no_incremental

Disables incremental optimization.

-shift_register_max_length integer

Specifies the maximum sequential length of the shift register for auto-identification.

Note: This option applies only when you specify -auto_identify_shift_register.

Default: Longest detected shift register segment

-shift_register_min_length integer

Specifies the minimum sequential length of the shift register for auto-identification.

Note: This option applies only when you specify -auto_identify_shift_register.

Default: 2

-spatial

Performs a quick placement to optimize the mapped gates.

Note: You must have access to the Encounter® place and route tool to run this command option.

to_clock_gated

Inserts clock-gating logic in an elaborated design, maps the clock-gating logic and performs RTL optimization on the rest of the design.

This option cannot be used with any of the following options: -to_generic, -to_mapped, -to_placed or -spatial.

Note: For clock-gating logic to be inserted, you must enable the lp_insert_clock_gating root attribute.

Note: Can only be used when you start from RTL. No clock-gating will be inserted if you start from a netlist.

-to_generic

Performs RTL optimization.

This is the default option if the RTL design has not been optimized yet.
**Note:** To avoid problems in the verification flow, it is recommended to use medium effort for generic synthesis. In Qor-critical designs, you may use high effort, but currently this can cause potential problems with the verification flow.

- **-to_mapped**
  Maps the specified design(s) to the cells described in the supplied technology library and performs logic optimization. The aim of the optimization is to provide the smallest possible implementation of the synthesized design that still meets the supplied timing goal.

  This is the default option when the design is in the generic or mapped state.

- **-to_placed**
  Performs placement and placement-based optimizations.

  This option is only available with the RTL Compiler Advanced Physical Option.

  **Note:** You must have access to the Encounter® place and route tool to run this command option.

  **Note:** This option requires an Encounter executable of version 8.1 or later. However, it is highly recommended that you use the same versions of Encounter and RTL Compiler.

**Examples**

The following example synthesizes and optimizes all of the top-level designs below the current position in the design hierarchy into generic logic.

```
rc:/> synthesize
```

The following table shows which actions are performed, depending on the state of the design.

**Table 7-1 Actions Performed with No Option Specified**

<table>
<thead>
<tr>
<th>Current Design State</th>
<th>Generic</th>
<th>Mapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTL Optimization</td>
<td>Mapping</td>
<td>Unmapping</td>
</tr>
<tr>
<td></td>
<td>Incremental</td>
<td>Remapping</td>
</tr>
<tr>
<td></td>
<td>Optimizations</td>
<td>Incremental</td>
</tr>
<tr>
<td></td>
<td>(same as -to_generic)</td>
<td>(same as -to_mapped)</td>
</tr>
</tbody>
</table>
The following example limits synthesis to a single design `main`:

```bash
rc:/> synthesize main
```

The following example maps multiple designs at the same time:

```bash
rc:/> synthesize -to_mapped design1 design2
```

After the following example, the design in memory will be at the Boolean (generic) level of abstraction:

```bash
rc:/> synthesize -to_generic
```

The following table shows the actions that are performed for the `-to_generic` option depending on the state of the design.

### Table 7-2 Actions Performed with `-to_generic` Option Specified

<table>
<thead>
<tr>
<th>Current Design State</th>
<th>RTL Optimization</th>
<th>Generic</th>
<th>Mapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL</td>
<td>nothing</td>
<td></td>
<td>Unmapping</td>
</tr>
</tbody>
</table>

The following example requests mapping of the design:

```bash
rc:/> synthesize -to_mapped
```

The following table illustrates how the `-to_mapped` option affects the design:

### Table 7-3 Actions Performed with `-to_mapped` Option Specified

<table>
<thead>
<tr>
<th>Current Design State</th>
<th>RTL Optimization</th>
<th>Generic</th>
<th>Mapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL</td>
<td>Mapping</td>
<td></td>
<td>Unmapping</td>
</tr>
<tr>
<td>Mapping</td>
<td>Incremental Optimizations</td>
<td>Remap</td>
<td></td>
</tr>
<tr>
<td>Incremental Optimizations</td>
<td>Incremental Optimizations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7-4 on page 381 illustrates how the -incremental option affects the design:

**Table 7-4  Actions Performed with -incremental Option Specified**

<table>
<thead>
<tr>
<th>Options</th>
<th>Current Design State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTL/Generic</td>
</tr>
<tr>
<td>-to_generic -incremental</td>
<td>Disable -incremental and proceed</td>
</tr>
<tr>
<td>-to_mapped -incremental</td>
<td>Disable -incremental and proceed</td>
</tr>
<tr>
<td>-incremental</td>
<td>Disable -incremental and proceed</td>
</tr>
</tbody>
</table>

The following command places and optimizes the design for silicon:

```
rc:/> synthesize -to_placed
```

The following table illustrates how the -to_placed option affects the design:

**Table 7-5  Actions Performed with -to_placed Option Specified**

<table>
<thead>
<tr>
<th>Options</th>
<th>Current Design State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTL Optimization</td>
</tr>
<tr>
<td>-to_placed</td>
<td>Mapping</td>
</tr>
<tr>
<td>-incremental</td>
<td>Placement</td>
</tr>
<tr>
<td></td>
<td>Post-placement incremental optimizations</td>
</tr>
<tr>
<td>-to_placed -incremental</td>
<td>(same as -to_placed)</td>
</tr>
</tbody>
</table>

**Related Information**

**Inserting Clock Gating** in *Low Power in Encounter RTL Compiler*

Affects these commands:  
- report area on page 425  
- report clock gating on page 432  
- report datapath on page 441
report design_rules on page 446
report gates on page 470
report power on page 507
report summary on page 535
report timing on page 541

Related Command
remove_assigns_without_optimization on page 366

Affected by these attributes:
auto_ungroup
iopt_force_constant_removal
iopt_sequential_resynthesis
iopt_sequential_resynthesis_min_effort
Hierarchical Flow

- `assemble_design` on page 384
- `generate_ilm` on page 385
- `read_ilm` on page 387
- `read_ilm_from_files` on page 388
assemble_design
assemble_design [design]

Integrates the interface logic models in the design.

Use this command after elaborating the design and reading in the DEF file for the top-level design (if physical ILM is read).

**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**

**design** Specifies the name of the design for which to do the checking. This argument is only required if multiple designs are loaded.

**Related Information**

Related commands: generate_ilm on page 385
read_ilm on page 387
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 RTL Compiler, 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.
-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 384
read_ilm on page 387
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 384
generate_ilm on page 385
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 RTL Compiler should write out
  the tool-generated ILM data.
- **-prefix string** Specifies the prefix for the ILM data generated by RTL Compiler.
  
  **Default:** the specified module name.
- **-sdc_by_mode string** Specifies a list of Tcl lists. Each Tcl 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.
Analysis and Report

- all_connected on page 392
- all_des on page 393
- all_des_inps on page 394
- all_des_insts on page 395
- all_des_outs on page 396
- all_des_segs on page 397
- all_lib on page 399
- all_lib_bufs on page 400
- all_lib_ties on page 401
- analyze_library_corners on page 402
- check_design on page 404
- clock_ports on page 409
- compare_sdc on page 410
- create_timing_bin on page 411
- fanin on page 413
- fanout on page 417
- report on page 420
- report_area on page 425
- report_boundary_opto on page 427
- report_case_analysis on page 429
- report_cdn_loop_breaker on page 430
- report cell_delay_calculation on page 431
- report clock_gating on page 432
- report clocks on page 438
- report congestion on page 440
- report datapath on page 441
- report design_rules on page 446
- report dft_chains on page 448
- report dft_clock_domain_info on page 453
- report dft_core_wrapper on page 454
- report dft_registers on page 458
- report dft_setup on page 462
- report dft_violations on page 466
- report disabled_transparent_latches on page 469
- report gates on page 470
- report hierarchy on page 474
- report instance on page 476
- report low_power_cells on page 479
- report low_power_intent on page 482
- report memory on page 486
- report memory_cells on page 487
- report messages on page 488
- report mode on page 490
- report multibit_inferencing on page 491
- report net_cap_calculation on page 496
- report net_delay_calculation on page 497
- report net_res_calculation on page 498
- report nets on page 499
■ report opcg_equivalents on page 502
■ report ple on page 503
■ report ple on page 503
■ report port on page 505
■ report power on page 507
■ report power_domain on page 519
■ report proto on page 521
■ report qor on page 523
■ report scan_compressibility on page 529
■ report sequential on page 531
■ report summary on page 535
■ report test_power on page 537
■ report timing on page 541
■ report units on page 552
■ report utilization on page 553
■ report yield on page 554
■ statistics on page 555
■ statistics add_metric on page 557
■ statistics log on page 558
■ statistics read on page 560
■ statistics remove_metric on page 561
■ statistics report on page 562
■ statistics reset on page 565
■ statistics run_stage_ids on page 566
■ statistics write on page 567
■ timestat on page 568
■ validate_timing on page 569
all_connected

all_connected {net [-leaf_pin] | 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

net

pin

pgpin

port

- 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.
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inps</td>
<td>Generates a list of all input ports of the specified clock or clock domain.</td>
</tr>
<tr>
<td>insts</td>
<td>Generates a list of all the instances in the design.</td>
</tr>
<tr>
<td>outs</td>
<td>Generates a list of all output ports of the specified clock or clock domain.</td>
</tr>
<tr>
<td>seqs</td>
<td>Generates a list of all the sequential instances in the design.</td>
</tr>
</tbody>
</table>

Related Information

Related commands:  
- all des inps on page 394  
- all des insts on page 395  
- all des outs on page 396  
- all des seqs on page 397
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.

  rc:/> 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]}
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.

  rc:/ 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
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.

      rc:/> 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
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.
Examples

- The following example returns all the sequential instances for the design named REID.
  ```
  rc:/> 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.
  ```
  rc:/> 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.
  ```
  rc:/> set_attribute map_to_master_slave_lssd true
  rc:/> set_attribute library jess.lib
  ...
  rc:/> all des seqs -master_slave

  /designs/summers/instances_seq/flop

  rc:/> 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`
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 400
all lib ties on page 401
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.

rc:/> all lib bufs
/libraries/slow/libcells/BUFX1 /libraries/slow/libcells/BUFX12
/libraries/slow/libcells/BUFX16 /libraries/slow/libcells/BUFX2
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.
  
  rc:/> all lib ties -hi
  
  /libraries/LUX/libcells/TIEHI /libraries/LUX/libcells/ANTENNA
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.
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
```

<table>
<thead>
<tr>
<th>Library filename</th>
<th>Average delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>/libraries/library_domains/set8/MF</td>
<td>170</td>
</tr>
<tr>
<td>/libraries/library_domains/set7/MS</td>
<td>176</td>
</tr>
<tr>
<td>/libraries/library_domains/set6/XF</td>
<td>100</td>
</tr>
<tr>
<td>/libraries/library_domains/set4/XS</td>
<td>352</td>
</tr>
<tr>
<td>/libraries/library_domains/set2/F</td>
<td>162</td>
</tr>
<tr>
<td>/libraries/library_domains/set5/S</td>
<td>193</td>
</tr>
<tr>
<td>/libraries/library_domains/set3/VF</td>
<td>115</td>
</tr>
<tr>
<td>/libraries/library_domains/set1/VS</td>
<td>335</td>
</tr>
</tbody>
</table>

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
BUFCSXAX   /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.
check_design

check_design [-lib_lef_consistency]
[-undriven [-threshold_fanout]] [-multidriven]
[-unloaded] [-unresolved] [-assigns]
[-constant [-threshold_fanout] [-through_tie_cell]]
[-preserved] [-physical_only]
[-report_scan_pins | -skip_scan_pins]
[-long_module_name] [-vname] [-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.
- 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.
- 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.
- long_module_name Reports subdesigns whose name exceeds 1.5K.
- multidriven  Reports ports and pins in the design that are multi-driven.
- 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_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.
**Examples**

The following example shows a sample report given when the command is executed without any options.

```
rc:/> check_design
       Checking the design.

       Check Design Report
-----------------------
Summary
-------
<table>
<thead>
<tr>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unresolved References</td>
<td>0</td>
</tr>
<tr>
<td>Empty Modules</td>
<td>0</td>
</tr>
<tr>
<td>Unloaded Port(s)</td>
<td>0</td>
</tr>
<tr>
<td>Unloaded Sequential Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Assigns</td>
<td>0</td>
</tr>
<tr>
<td>Undriven Port(s)</td>
<td>0</td>
</tr>
<tr>
<td>Undriven Leaf Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Undriven hierarchical pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Multidriven Port(s)</td>
<td>0</td>
</tr>
<tr>
<td>Multidriven Leaf Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Multidriven hierarchical Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Multidriven unloaded net(s)</td>
<td>0</td>
</tr>
<tr>
<td>Constant Port(s)</td>
<td>0</td>
</tr>
<tr>
<td>Constant Leaf Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Constant hierarchical Pin(s)</td>
<td>0</td>
</tr>
<tr>
<td>Preserved leaf instance(s)</td>
<td>5</td>
</tr>
<tr>
<td>Preserved hierarchical instance(s)</td>
<td>1</td>
</tr>
<tr>
<td>Libcells with no LEF cell</td>
<td>601</td>
</tr>
<tr>
<td>Physical (LEF) cells with no libcell</td>
<td>846</td>
</tr>
</tbody>
</table>

Done Checking the design.
```
The following example reports the preserved instances in your design.

```
rc:/> 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.

The following command reports the cells which are only in .lib or in the physical library (LEF files).

```
rc:/> 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_SMDFFHQQX8
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).

```
rc:/> 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.
```

The following command reports all undriven pins excluding the DFT-related pins.

```
rc:/> 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.

```
rc:/> 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.
```

Related Information

Run DFT Rule Checker and Floating Nets and X-Source Checks in Design for Test in Encounter RTL Compiler
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.
  
  ```
  rc:/> 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.
  
  ```
  rc:/> define_clock -period 3000 -name clock1 [clock_ports]
  ```

**Related Information**

Affected by this command: `define_clock` on page 318
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 Encounter® Conformal® Constraint Designer (CCD) software.

Options and Arguments

-design string Specifies the top-level design in RTL Compiler.
-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

Comparing SDC Constraint Files in Interfacing between Encounter RTL Compiler and Encounter Conformal

Related command: write_do_ccd compare_sdc on page 247
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 RTL Compiler commands (or their SDC equivalent): multi_cycle, path_adjust, path_delay, path_disable, or path_group.

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.

Related Information

Creates these attributes  
Timing Bin Attributes  
Timing Path Attributes
fanin

fanin [pin | pgpin | port | subport | net]...
   [-min_logic_depth integer]
   [-max_logic_depth integer]
   [-min_pin_depth integer] [-max_pin_depth integer]
   [-structural [-timing_model_comb] [-floating_subports]]
   [-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.

- max_logic_depth integer

Specifies the maximum number of logic levels to go back to report the fanin cone information.

   **Default:** Infinity

- 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 | 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.

-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.
Examples

Consider the design below.

The following example returns all the pins in the fanin cone for port `out`:

```
rc:/> 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.

```
rc:/> 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`.

```
rc:/> 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 9-1 on page 416.

```
rc:/> 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.

```
rc:/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
```
Use the `ls -dir` command with the redirect arrow to redirect the output to the specified file.

```
rc:/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 9-1, but without using the `-startpoint` option.

```
rc:/> fanin S
```

In this case, the command returns in addition to the input port IN and clock pin CK, pins A and Q.

**Figure 9-1 Example Design for fanin**
fanout

fanout {pin | pgpin | port | subport | net}...
   [-min_logic_depth integer]
   [-max_logic_depth integer]
   [-min_pin_depth integer] [-max_pin_depth integer]
   [-structural [-timing_model_comb] [-floating_subports]]
   [-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.

- max_logic_depth integer
  Specifies the maximum number of logic levels to go back to report the fanout cone information.

  Default: Infinity

- 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.

-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.

Examples

The following example returns the pins in the fanout cone of port en in the design below.
The following example executes a path disable on all the endpoints to which reg1/CK fans out.

```bash
rc:/> 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.

```bash
rc:/> 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.

```bash
rc:/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.

```bash
rc:/designs/malexander/ports_in> ls -dir [fanout in1[0]] > malexander.txt
```

You can also append arrows (">>").
**Command Reference for Encounter RTL Compiler**  
Analysis and Report

---

**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, RTL Compiler 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 (RC-P flow)

When running a physical flow, RTL Compiler 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
```

**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.

- **dftviolations**
  Reports the DFT violations.

- **disabled_transparent_latches**
  Reports disabled transparent latches.

- **gates**
  Generates a gates report.

- **hierarchy**
  Reports the design hierarchy information.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>Generates a report on the instances of the current design.</td>
</tr>
<tr>
<td>low_power_cells</td>
<td>Reports information about the low power cells inserted in the design.</td>
</tr>
<tr>
<td>low_power_intent</td>
<td>Prints out the power intent included in the power intent file that was read in.</td>
</tr>
<tr>
<td>memory</td>
<td>Reports the memory usage in the compilation environment.</td>
</tr>
<tr>
<td>memory_cells</td>
<td>Reports the memory cells in the library.</td>
</tr>
<tr>
<td>messages</td>
<td>Generates a summary of error messages that have been issued.</td>
</tr>
<tr>
<td>mode</td>
<td>Reports the active and inactive modes of an instance.</td>
</tr>
<tr>
<td>multibit_inferencing</td>
<td>Reports on multibit cells in the design or library.</td>
</tr>
<tr>
<td>net_cap_calculation</td>
<td>Reports how the capacitance of the net is calculated.</td>
</tr>
<tr>
<td>net_delay_calculation</td>
<td>Reports how the net delay is calculated.</td>
</tr>
<tr>
<td>net_res_calculation</td>
<td>Reports how the resistance of the net is calculated.</td>
</tr>
<tr>
<td>nets</td>
<td>Generates a report on the nets of the current design.</td>
</tr>
<tr>
<td>opcg_equivalents</td>
<td>Reports the OPCG-equivalency mappings.</td>
</tr>
<tr>
<td>ple</td>
<td>Reports physical layout estimation data</td>
</tr>
<tr>
<td>port</td>
<td>Generates reports on the ports of the current design.</td>
</tr>
<tr>
<td>power</td>
<td>Generates a power leakage report.</td>
</tr>
<tr>
<td>power_domain</td>
<td>Generates a report with power domain information.</td>
</tr>
<tr>
<td>proto</td>
<td>Generates a report with prototype synthesis information.</td>
</tr>
<tr>
<td>qor</td>
<td>Generates a QoR report.</td>
</tr>
<tr>
<td>scan_compressibility</td>
<td>Reports the scan compressibility of a design.</td>
</tr>
<tr>
<td>sequential</td>
<td>Generates a report on the sequential elements of the design.</td>
</tr>
<tr>
<td>slew_calculation</td>
<td>Reports how the slew on the driver pin of a libcell instance is calculated.</td>
</tr>
<tr>
<td>summary</td>
<td>Generates an area, timing, and design rule violations report.</td>
</tr>
</tbody>
</table>
test_power  Reports estimated power of design during scan test

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 425
report boundary_opto on page 427
report case_analysis on page 429
report cdn_loop_breaker on page 430
report cell_delay_calculation on page 431
report clock_gating on page 432
report clocks on page 438
report congestion on page 440
report datapath on page 441
report design_rules on page 446
report dft_chains on page 448
report dft_clock_domain_info on page 453
report dft_core_wrapper on page 454
report dft_registers on page 458
report dft_setup on page 462
report dft_violations on page 466
report disabled_transparent_latches on page 469
report gates on page 470
report hierarchy on page 474
report instance on page 476
report memory on page 486
report memory_cells on page 487
report messages on page 488
report mode on page 490
report multibit_inferencing on page 491
report net_cap_calculation on page 496
report net_delay_calculation on page 497
report net_res_calculation on page 498
report nets on page 499
report opcg_equivalents on page 502
report ple on page 503
report port on page 505
report power on page 507
report power_domain on page 519
report proto on page 521
report qor on page 523
report scan_compressibility on page 529
report sequential on page 531
report slew_calculation on page 534
report summary on page 535
report test_power on page 537
report timing on page 541
report units on page 552
report utilization on page 553
report yield on page 554
report area

report area
[-depth integer [-instance_hierarchy instance]]
[-min_count integer] [-physical]
[-summary] [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.

file
  Specifies the name of the file to which to write the report.

-instance_hierarchy 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.

-physical
  Specifies to use the LEF cell width and height values to calculate the area.

-summary
  Prints the area summary for the design.
Examples

- The following example generates the area report for the current design.

```shell
rc:/> report area
```

*Generated by:* RTL Compiler (RC) 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

- The following command prints the area summary.

```shell
rc:/> report area -summary
```

```
************* 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.

```shell
rc:/> report area -instance_hier ma1 -depth 2
```

```
************* Area Summary *************
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

Analyzing the Results in Simple PLE Flow, Spatial Flow, and RC-P Flow in Design with RTL Compiler Physical

Affected by this command: `synthesize` on page 377

Affected by this attribute: `shrink_factor`
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.

rc:/> report boundary_opto
========================================================================
... 
========================================================================

<table>
<thead>
<tr>
<th>Instance</th>
<th>Pin</th>
<th>Boundary Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst</td>
<td>out1</td>
<td>routed opposite signal through 'inst/a' around 'inst'</td>
</tr>
<tr>
<td></td>
<td>out2</td>
<td>pushed opposite signal through 'inst/out'</td>
</tr>
</tbody>
</table>
Related Information

Related attribute:         boundary_change
report case_analysis

report case_analysis  
   design [> file]

Reports the user-set case constants on pins and ports.

Options and Arguments

    design            Specifies the design for which you want to generate a report.
    file              Reports the case constants.

Example

   ------------------------------------------
   report case_analysis
   ------------------------------------------
   Pin Name Case
   -----------
   OR1/A     0
   en        1

Related Information

Set by this command:        dc::set_case_analysis
Set by these attributes:   (pin) timing_case_logic_value
                           (port) timing_case_logic_value
**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*.

```
rc://> 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
```

**Related Information**

Related command: `remove_cdn_loop_breaker` on page 1082
**report cell_delay_calculation**

```
report cell_delay_calculation
  -from_pin pin -to_pin pin
  [-from_rise] [-from_fall]
  [-to_rise] [-to_fall] [> 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.

- **-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.
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]]
    [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.

- **-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.

-instance hier_instance

Prints the clock-gating report for the specified hierarchical instance.

-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.

-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.

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.

```
rc:/> 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
...      ...                       ...     ...                ...
alu      RC_CG_HIER_INST0       RC      8    aluout_reg[6]        100.000
alu      RC_CG_HIER_INST0       RC      8    aluout_reg[7]        100.000
-------------------------------------------------------------------------------------
Total 1 8
-------------------------------------------------------------------------------------
```

- The following command gives the reason why the flip-flops remain ungated. In this case, there is no return value.

```
rc:/> 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.

```
rc:/> report clock_gating -fanout_histogram
============================================================
...
============================================================
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.

```
rc:/> report clock_gating -fanout_histogram -step {{1 6} {7 15}}
```

---

**CG Fanout Histogram**

<table>
<thead>
<tr>
<th>Fanout-Size</th>
<th>Num-CGs</th>
<th>Total-FFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7 to 15</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>16 and higher</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The following command generates detailed clock-gating information for the specified clock-gating instance.

```
rc:/> 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/en) 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**:

<table>
<thead>
<tr>
<th>Module</th>
<th>Clock Gating Instance</th>
<th>Origin</th>
<th>Fanout</th>
<th>Gated Flip-flops</th>
<th>Register Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu</td>
<td>RC_CG_HIER_INST0</td>
<td>RC</td>
<td>8</td>
<td>100.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aluout_reg[0]</td>
<td>100.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aluout_reg[1]</td>
<td>100.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aluout_reg[6]</td>
<td>100.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aluout_reg[7]</td>
<td>100.000</td>
</tr>
</tbody>
</table>

**Total**: 1 8
The following command reports the number of flip-flops that are clock gated by the specified clock.

```
rc:/> 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.

```
rc:/> 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
------------------------------------------
rc:/> 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

**Reporting Clock-Gating Information** in *Low Power in Encounter RTL Compiler*

**Clock Gating Cell Specification** in the *Library Guide for Encounter RTL Compiler*.

Affected by this command: `synthesize` on page 377

Affected by these attributes:
- `lp_clock_gating_add_obs_port`
- `lp_clock_gating_add_reset`
- `lp_clock_gating_cell`
- `lp_clock_gating_control_point`
- `lp_clock_gating_exclude`
- `lp_clock_gating_style`
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.

rc:/> report clocks -generated
============================================================================
Generated by:                         RTL Compiler (RC) Version
Generated on:                         Date
Module:                               test
Technology library:                  tutorial 1.0
Operating conditions:                typical_case (balanced_tree)
Wireload mode:                        enclose
============================================================================
Clock Description
-------------------
Clock       Period    Rise   Fall    Domain   Pin/Port Registers
------------- ----------- ------- ------- --------- -------------
CLK1        4000.0     0.0     2000.0  domain_1  Clk             5
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

<table>
<thead>
<tr>
<th>Clock Name</th>
<th>Clock Latency</th>
<th>Unknown Source Latency</th>
<th>Setup Uncertainty</th>
<th>Unknown Setup Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK1</td>
<td>140.0</td>
<td>150.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>CLK2</td>
<td>120.0</td>
<td>120.0</td>
<td>110.0</td>
<td>110.0</td>
</tr>
<tr>
<td>CLK3</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>CLK4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Clock Relationship (with uncertainty & latency)

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>R-&gt;R</th>
<th>R-&gt;F</th>
<th>F-&gt;R</th>
<th>F-&gt;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK1</td>
<td>CLK1</td>
<td>3900.0</td>
<td>1900.0</td>
<td>3900.0</td>
<td></td>
</tr>
<tr>
<td>CLK1</td>
<td>CLK2</td>
<td>1840.0</td>
<td>840.0</td>
<td>1840.0</td>
<td>840.0</td>
</tr>
<tr>
<td>CLK2</td>
<td>CLK3</td>
<td>2900.0</td>
<td>1400.0</td>
<td>2900.0</td>
<td></td>
</tr>
<tr>
<td>CLK2</td>
<td>CLK4</td>
<td>2800.0</td>
<td>1600.0</td>
<td>2800.0</td>
<td></td>
</tr>
<tr>
<td>CLK3</td>
<td>CLK4</td>
<td>6000.0</td>
<td>3000.0</td>
<td>3000.0</td>
<td>6000.0</td>
</tr>
</tbody>
</table>

Related Information

See the following sections in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Reporting Clocks

Generating Clock Reports

Analyzing and Reporting Multi-Mode Information

Affected by this command: `create_mode` on page 315
**report congestion**

`report congestion [ >file]`

Reports the total number (and 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`.

```
rc:/> report congestion
==========================================
Generated by: RTL Compiler (RC) 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

<table>
<thead>
<tr>
<th>Item</th>
<th>Oflow/Avail</th>
<th>Gcell (Bounding-box location)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Overflow</td>
<td>-111/201</td>
<td>20,18 (480.0,432.0),(504.0,456.0)</td>
</tr>
<tr>
<td>Max. Overflow (H)</td>
<td>-52/118</td>
<td>31,29 (744.0,696.0),(768.0,720.0)</td>
</tr>
<tr>
<td>Max. Overflow (V)</td>
<td>-75/87</td>
<td>19,17 (456.0,408.0),(480.0,432.0)</td>
</tr>
<tr>
<td>Total Overflow</td>
<td>-7172</td>
<td></td>
</tr>
<tr>
<td>Total Overflow (H)</td>
<td>-1329</td>
<td></td>
</tr>
<tr>
<td>Total Overflow (V)</td>
<td>-5843</td>
<td></td>
</tr>
</tbody>
</table>
```

**Related Information**

Affected by this command: `synthesize -to_placed`
**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, RTL Compiler 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, RTL Compiler 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.
-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.
Examples

- The following example generates a report of the datapath components for the `rpdp1_basic` design.

```
rc:/> 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
    dl 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.

```
Type CellArea Percentage
-------------------------------------
datapath modules 4938.00 100.00
mux modules 0.00 0.00
others 0.00 0.00
-------------------------------------
total 4938.00 100.00
```
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.

  rc:/> report datapath -sort area

  ==============================================================
  ... 
  ==============================================================

Inferred components

<table>
<thead>
<tr>
<th>Operator</th>
<th>Signedness</th>
<th>Inputs</th>
<th>Outputs</th>
<th>CellArea</th>
<th>Line</th>
<th>Col</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>lt_leq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_14_26</td>
<td></td>
<td>signed</td>
<td>4x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>module:add_signed</td>
<td>very_fast</td>
<td>+</td>
<td>signed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31.50</td>
<td>14</td>
<td>26</td>
<td>lt_leq.v</td>
</tr>
<tr>
<td>lt_leq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lt_13_25</td>
<td></td>
<td>signed</td>
<td>4x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>module:lt_signed</td>
<td>very_fast</td>
<td>&lt;</td>
<td>signed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26.25</td>
<td>13</td>
<td>25</td>
<td>lt_leq.v</td>
</tr>
<tr>
<td>lt_leq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lt_16_25</td>
<td></td>
<td>unsigned</td>
<td>4x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>module:lt_unsigned</td>
<td>very_fast</td>
<td>&lt;</td>
<td>unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26.25</td>
<td>16</td>
<td>25</td>
<td>lt_leq.v</td>
</tr>
<tr>
<td>lt_leq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>le_17_26</td>
<td></td>
<td>unsigned</td>
<td>4x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>module:leq_unsigned</td>
<td>very_fast</td>
<td>&lt;=</td>
<td>unsigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.50</td>
<td>17</td>
<td>26</td>
<td>lt_leq.v</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>CellArea</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapath modules</td>
<td>109.50</td>
<td>75.26</td>
</tr>
<tr>
<td>external muxes</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>others</td>
<td>36.00</td>
<td>24.74</td>
</tr>
<tr>
<td>total</td>
<td>145.50</td>
<td>100.00</td>
</tr>
</tbody>
</table>
The following command sorts the report first by slack, then by area.

rc:/> report datapath -sort {slack area}
=============================================================================

Inferred components

<table>
<thead>
<tr>
<th>Operator</th>
<th>Signedness</th>
<th>Inputs</th>
<th>Outputs</th>
<th>CellArea</th>
<th>Line</th>
<th>Col</th>
<th>Filename</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>lt_leq</td>
<td>&lt;</td>
<td>signed</td>
<td></td>
<td>26.25</td>
<td>13</td>
<td>25</td>
<td>lt_leq.v</td>
<td>-145.2</td>
</tr>
<tr>
<td>lt_leq</td>
<td>&lt;</td>
<td>unsigned</td>
<td></td>
<td>26.25</td>
<td>16</td>
<td>25</td>
<td>lt_leq.v</td>
<td>-145.2</td>
</tr>
<tr>
<td>leq</td>
<td>&lt;=</td>
<td>unsigned</td>
<td></td>
<td>25.50</td>
<td>17</td>
<td>26</td>
<td>lt_leq.v</td>
<td>-53.3</td>
</tr>
<tr>
<td>add</td>
<td>+</td>
<td>signed</td>
<td></td>
<td>31.50</td>
<td>14</td>
<td>26</td>
<td>lt_leq.v</td>
<td>49.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>CellArea</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapath modules</td>
<td>109.50</td>
<td>75.26</td>
</tr>
<tr>
<td>external muxes</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>others</td>
<td>36.00</td>
<td>24.74</td>
</tr>
<tr>
<td>total</td>
<td>145.50</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Related Information

Datapath Reporting in Datapath Synthesis in Encounter RTL Compiler

Affected by this attribute: hdl_track_filename_row_col
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: RTL Compiler (RC) 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.

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: RTL Compiler (RC) version ....
```

```
Cells with no max capacitance lut table (total=6)
Cell Library # of instances
-----------------------------------
AND2X2 mylib 1
BUFX12 mylib 1
BUFX16 mylib 2
DFFHQL4 mylib 20002
INVX16 mylib 1
NAND2XL mylib 10001

Instances with frequency constraint outside cell's max capacitance lut's range (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

Setting Design Rule Constraints in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Affected by this command: synthesize on page 377

Affected by these attributes: ignore_library_max_fanout
max_capacitance
max_fanout
max_transition
use_max_cap_lut
report dft_chains

report dft_chains [design] [-chain 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 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.

- **-dft_configuration_mode dft_configuration_mode_name**
  
  Reports the scan chains related to the specified scan mode.
Examples

The following example shows one scan chain, with three user-defined segments.

```
rc:/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.

```
rc:/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
```

**file**

Specifies the name of the file to which to write the report.

**-opcg_scan_side**

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.
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 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.
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_4 (sdi: g121/SWBOX_SI[4])>
    bit 9  out_reg[19] <clk (rise)>
    bit 10 out_reg[20] <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
The following command reports two side scan chains, each with one OPCG segment.

```
rc:/> report dft_chains -opcg_side_scan
```

Reporting 2 side scan chains
Side-scan chain 1:
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

See the following sections in Design for Test in Encounter RTL Compiler

- Reporting on All Scan Chains
- Compressing Scan Chains
- Concatenating Scan Chains

Affected by this command: connect_scan_chains on page 681
compress_scan_chains on page 660

Related attributes: Actual Scan Chain attributes
Actual Scan Segment attributes
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.

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.

Related Information

Inserting On-Product Clock Generation Logic in Design for Test in Encounter RTL Compiler

Related command:  insert_dft_opcg on page 825
**report dft_core_wrapper**

```
report dft_core_wrapper
    [-wrapped | -unwrapped]
    [-inside_core] [-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.

- `-inside_core` Reports the wrapper cells inside the core instance.

- `-location` 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.
Examples

The following examples illustrate the different report options on the same design.

```shell
rc:/> report dft_core_wrapper test
```

Reporting 13 ports/pins

<table>
<thead>
<tr>
<th>Port/Pin</th>
<th>type</th>
<th>usage</th>
<th>wrapper_objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEXT</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WIG</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WINT</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WOG</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WSEN_in</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>clk</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WSEN_out</td>
<td>shift_enable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>se</td>
<td>shift_enable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>output</td>
<td>wrap_4</td>
</tr>
<tr>
<td>in2</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>in2</td>
<td>shared</td>
<td>output</td>
<td>wrap_4</td>
</tr>
<tr>
<td>in3</td>
<td>shared</td>
<td>input</td>
<td>wrap_3</td>
</tr>
<tr>
<td>out1</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>out1</td>
<td>shared</td>
<td>input</td>
<td>wrap_4</td>
</tr>
<tr>
<td>out2</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
</tbody>
</table>

**Wrapper insertion summary report**

<table>
<thead>
<tr>
<th>Number of Dedicated wrapper cells: 0</th>
<th>input bounding: 0</th>
<th>output bounding: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Shared wrapper cells: 4</td>
<td>input bounding: 3</td>
<td>output bounding: 1</td>
</tr>
<tr>
<td>Total no. of wrapper cells: 4</td>
<td>input bounding: 3</td>
<td>output bounding: 1</td>
</tr>
</tbody>
</table>

```shell
rc:/> report dft_core_wrapper test -wrapped
```

Reporting 13 ports/pins

<table>
<thead>
<tr>
<th>Port/Pin</th>
<th>type</th>
<th>usage</th>
<th>wrapper_objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>in1</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>output</td>
<td>wrap_4</td>
</tr>
<tr>
<td>in2</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>in2</td>
<td>shared</td>
<td>output</td>
<td>wrap_4</td>
</tr>
<tr>
<td>in3</td>
<td>shared</td>
<td>input</td>
<td>wrap_3</td>
</tr>
<tr>
<td>out1</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
<tr>
<td>out1</td>
<td>shared</td>
<td>input</td>
<td>wrap_4</td>
</tr>
<tr>
<td>out2</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
</tr>
</tbody>
</table>

**Wrapper insertion summary report**

<table>
<thead>
<tr>
<th>Number of Dedicated wrapper cells: 0</th>
<th>input bounding: 0</th>
<th>output bounding: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Shared wrapper cells: 4</td>
<td>input bounding: 3</td>
<td>output bounding: 1</td>
</tr>
<tr>
<td>Total no. of wrapper cells: 4</td>
<td>input bounding: 3</td>
<td>output bounding: 1</td>
</tr>
</tbody>
</table>

```shell
rc:/> report dft_core_wrapper -report_flops
```

Reporting 13 ports/pins

<table>
<thead>
<tr>
<th>Port/Pin</th>
<th>type</th>
<th>Usage</th>
<th>wrapper_objects</th>
<th>wrapper_flops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEXT</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td>tmp5_reg</td>
</tr>
<tr>
<td>WIG</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td>tmp4_reg</td>
</tr>
<tr>
<td>WINT</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>WOG</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>WSEN_in</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>clk</td>
<td>wrapper_control</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>WSEN_out</td>
<td>shift_enable</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>se</td>
<td>shift_enable</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>input</td>
<td>wrap_1</td>
<td>tmp1_reg</td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>input</td>
<td>wrap_4</td>
<td></td>
</tr>
<tr>
<td>in1</td>
<td>shared</td>
<td>output</td>
<td>wrap_3</td>
<td></td>
</tr>
</tbody>
</table>
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

rc:/> 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

The following reports shows the wrapper cells inside the core instance i_core1. In the first report, wrapper cells were inserted on the input list. In the second report, wrapper cells were also inserted on the output list.

rc:/> report dft_core_wrapper -inside_core i_core1
first report:
Reporting 15 ports/pins
Port/Pin                  type          usage   wrapper_objects
---------------          -------          -------          ---------------
  i_core1/se            wrapper_control - -          -
  i_core1/wclk          wrapper_control - -          -
  i_core1/wext          wrapper_control - -          -
  i_core1/wint          wrapper_control - -          -
  i_core1/tclk          test_clock - -              -
  i_core1/tm            test_control - -            -
  i_core1/out[0]        not processed - -          -
  i_core1/out[1]        not processed - -          -
  i_core1/out[2]        not processed - -          -
  i_core1/out[3]        not processed - -          -
  i_core1/in[0]         shared input            wrap_in_3
  i_core1/in[1]         shared input            wrap_in_4
  i_core1/in[2]         shared input            wrap_in_1
  i_core1/in[3]         shared input            wrap_in_2

Wrapper insertion summary report
-----------------------------------------------------------------------------------------------
Number of Dedicated wrapper cells :0 input bounding :0 output bounding :0
Number of Shared wrapper cells :4 input bounding :4 output bounding :0
-----------------------------------------------------------------------------------------------
Total no. of wrapper cells :4 input bounding :4 output bounding :0

second report:
Reporting 15 ports/pins
Port/Pin                  type          usage   wrapper_objects
---------------          -------          -------          ---------------
  i_core1/se            wrapper_control - -          -
  i_core1/wclk          wrapper_control - -          -
  i_core1/wext          wrapper_control - -          -
  i_core1/wint          wrapper_control - -          -
  i_core1/tclk          test_clock - -              -
  i_core1/tm            test_control - -            -
  i_core1/out[0]        excluded - -              -
  i_core1/out[1]        not processed - -          -
  i_core1/out[2]        not processed - -          -
  i_core1/out[3]        not processed - -          -
  i_core1/in[0]         shared input            wrap_in_3
  i_core1/in[1]         shared input            wrap_in_4
  i_core1/in[2]         shared input            wrap_in_1
  i_core1/in[3]         shared input            wrap_in_2
Wrapper insertion summary report

Number of Dedicated wrapper cells :0  input bounding :0  output bounding :0
Number of Shared wrapper cells :4  input bounding :4  output bounding :0
Total no. of wrapper cells :4  input bounding :4  output bounding :0

Related Information

Affected by this command: insert_dft_wrapper_cell on page 861
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.
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.
Example

The following example shows that 1 flip-flop passed the DFT rule checks, while 4 flip-flops failed the tests.

rc:/> 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.

rc:/> 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
...

Reporting registers that fail DFT rules
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
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

See the following sections in *Design for Test in Encounter RTL Compiler*

- Reporting Status on All Flip-Flops
- Controlling Mapping to Scan Flip-Flops

Affected by this command: check_dft_rules on page 648
Related attributes:  
  dft_dont_scan  
  dft_scan_style  
  dft_status  
  dftViolation  
  preserve (instance)  
  preserve (subdesign)  

Scan Segment Attributes
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.

rc:/> 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


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.

rc:/designs/test> report dft_setup

Design Name

Scan Style

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
  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:
  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 Design for Test in Encounter RTL Compiler

Affected by these constraints:
  define_dft shift_enable on page 751
  define_dft test_clock on page 762
  define_dft test_mode on page 766

Affected by this command:
  check_dft_rules on page 648
  connect_scan_chains on page 681

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
  Scan Chain Attributes
  Test Clock Attributes
  Test Signal Attributes
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 to get meaningful information.

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.
Examples

- The following example shows that the design has four violations, but only detailed information on the clock violations is requested.

  ```
  rc:/> 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: gl/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.

  ```
  rc:/> 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
  ```
Violation Rule Summary Report

[TRISTATE_NET-01] tristate net potentially driven by conflicting values : 1

Related Information

Reporting the DFT Violations in *Design for Test in Encounter RTL Compiler*

Affected by this command:
- check_dft_rules on page 648
- fix_dft_violations on page 772

Related attributes:
- Violations Attributes
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 78
**report gates**

```plaintext
report gates [-library_domain library_domain_list]
   [-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.

- `power` Adds leakage power and internal power information.

- `yield` Reports the yield cost and yield percentage values for each library cell.
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.

```
rc:// report gates
============================================================
... Module: alu
Technology library: tutorial 1.0
Operating conditions: typical_case (balanced_tree)
Wireload mode: enclosed
============================================================
Gate Instances Area Library
---------------------------------------
flopdr 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
```
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.

```
rc:/> report gates -power
rc:/> report gates -power
============================================================
...  
...  
...  
Gate       Instances   Area     Leakage Power (nW) Internal Power (nW) Library
---------------------------------------------------------------
ACHCONX2TH  1    28.856  59.948    372.198 slow_hvt
ADDPHX1     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 
---------------------------------------------------------------
Type       Instances   Area     Area % Power (nW) Power %  Leakage Leakage Internal Internal
------------------------------------------------------------------------------
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 94809.978 87.0 114828.979 82.2  366672.348 72.5
------------------------------------------------------------------------------
total      11980 109000.238 100.0 139743.990 100.0  505460.121 100.0
```
The following example reports the yield cost and yield percentage values for each library cell.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Instances</th>
<th>Area</th>
<th>Cost</th>
<th>Yield</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>flopdrs</td>
<td>33</td>
<td>264.000</td>
<td>3.39278e-06</td>
<td>99.9997</td>
<td>tutorial</td>
</tr>
<tr>
<td>inv1</td>
<td>103</td>
<td>51.500</td>
<td>1.5022e-06</td>
<td>99.9998</td>
<td>tutorial</td>
</tr>
<tr>
<td>nand2</td>
<td>315</td>
<td>315.000</td>
<td>1.08311e-05</td>
<td>99.9989</td>
<td>tutorial</td>
</tr>
<tr>
<td>nor2</td>
<td>19</td>
<td>28.500</td>
<td>6.79989e-07</td>
<td>99.9999</td>
<td>tutorial</td>
</tr>
<tr>
<td>total</td>
<td>470</td>
<td>659.000</td>
<td>1.64061e-05</td>
<td>99.9984</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Instances</th>
<th>Area</th>
<th>Area %</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential</td>
<td>33</td>
<td>264.000</td>
<td>40.1</td>
</tr>
<tr>
<td>inverter</td>
<td>103</td>
<td>51.500</td>
<td>7.8</td>
</tr>
<tr>
<td>logic</td>
<td>334</td>
<td>343.500</td>
<td>52.1</td>
</tr>
<tr>
<td>total</td>
<td>470</td>
<td>659.000</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Related Information

Generating a Gates Report in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Reporting Power Consumption per Used Cell Types in Low Power in Encounter RTL Compiler

Affected by this command: **synthesize** on page 377
**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 hierarchy 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 hierarchy 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 )
```
Related Information

Affected by this attribute: hdl_track_filename_row_col
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.

rc:/> report instance -timing n0000D3666

... Module:  dp1dalign ...

============================================================
Instance Timing Info
============================================================
Instance n0000D3666 of libcell nor2

<table>
<thead>
<tr>
<th>Arc</th>
<th>Arc</th>
<th>Slew</th>
<th>Slew</th>
<th>From To</th>
<th>In</th>
<th>Load</th>
<th>Out</th>
<th>Delay</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>f</td>
<td>46.2</td>
<td>20.7</td>
<td>57.2</td>
<td>136.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Y</td>
<td>r</td>
<td>46.2</td>
<td>20.7</td>
<td>57.2</td>
<td>136.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>f</td>
<td>16.5</td>
<td>20.7</td>
<td>57.2</td>
<td>134.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>r</td>
<td>16.5</td>
<td>20.7</td>
<td>57.2</td>
<td>134.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

November 2015  476  Product Version 14.2
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 

<table>
<thead>
<tr>
<th>Arc</th>
<th>Arc</th>
<th>Slew</th>
<th>Slew</th>
<th>From</th>
<th>To</th>
<th>In</th>
<th>Load</th>
<th>Out</th>
<th>Delay</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK</td>
<td>Q</td>
<td>0.0</td>
<td>10.0</td>
<td>173.9</td>
<td>387.0</td>
<td>inf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CK</td>
<td>QN</td>
<td>0.0</td>
<td>0.0</td>
<td>64.3</td>
<td>inf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Instance Power Info 
---

Instance m2/o_m2_clk2_1_reg_0 of Libcell EDFFX1 

<table>
<thead>
<tr>
<th>Leakage</th>
<th>Internal</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power(nW)</td>
<td>Power(nW)</td>
<td>Power(nW)</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>32.822</td>
<td>20652.925</td>
<td>182.250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin</th>
<th>Net</th>
<th>Computed Probability</th>
<th>Computed Toggle Rate(/ns)</th>
<th>Net Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>o_m2_clk2_1[0]</td>
<td>0.6</td>
<td>0.0312</td>
<td>182.250</td>
</tr>
<tr>
<td>CK</td>
<td>n_0</td>
<td>0.5</td>
<td>1.8724</td>
<td>4913.916</td>
</tr>
<tr>
<td>D</td>
<td>in_2[21]</td>
<td>0.5</td>
<td>0.0208</td>
<td>138.510</td>
</tr>
<tr>
<td>E</td>
<td>en_2[7]</td>
<td>0.5</td>
<td>0.0208</td>
<td>473.850</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arc</th>
<th>Arc</th>
<th>Activity(/ns)</th>
<th>Energy(fJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK</td>
<td>CK</td>
<td>!D &amp; !E</td>
<td>0.4681</td>
</tr>
<tr>
<td>CK</td>
<td>CK</td>
<td>D &amp; !E</td>
<td>0.4681</td>
</tr>
<tr>
<td>CK</td>
<td>CK</td>
<td>!D &amp; E</td>
<td>0.4681</td>
</tr>
<tr>
<td>CK</td>
<td>CK</td>
<td>D &amp; E</td>
<td>0.4681</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td></td>
<td>0.0208</td>
</tr>
<tr>
<td>CK</td>
<td>CK</td>
<td>D &amp; E</td>
<td>0.4681</td>
</tr>
<tr>
<td>CK</td>
<td>Q</td>
<td></td>
<td>0.0312</td>
</tr>
</tbody>
</table>
The following example shows a detailed instance report.

```plaintext
rc:/> 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
-----------------------------------------------

Related Information

Generating Cell Instance Reports in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

Reporting Power on Leaf Instances in Low Power in Encounter RTL Compiler
report low_power_cells

report low_power_cells [-design design] [-instance Instance] [-lp_instance instance] [-isolation_only | -level_shifter_only | -combo_only | -state_retention_only] [-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.
- **-state_retention_only** Reports state retention cells only.
- **-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.

```
rc:/> report low_power_cells -summary
============================================================
| Generated by: RTL Compiler-D (RC) 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.

```
rc:/> report low_power_cells -design top -level_shifter_only
============================================================
| Generated by: Encounter(R) RTL Compiler 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.

```
rc:/> report low_power_cells -design top -detail
============================================================
| Generated by: Encounter(R) RTL Compiler version |
...|
============================================================

Level Shifter Report
Design/Instance top
Rule : pd2.ls_self_hiconn Count : 2
ul/in (pd1 -> pd1) pd2_ls_self_hiconn_0 (LVLHLD8BWPHVT)
ul/out (pd1 -> pd2) pd2_ls_self_hiconn_1 (LVLHLD8BWPHVT)
Total Level Shifter: 2
```
The following report shows detailed information about one level shifter instance.

```
rc:/> report low_power_cells -design top -lp_instance pd2_ls_self_hiconn_1
============================================================
Generated by:       Encounter(R) RTL Compiler 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

See the following chapters in *Low Power in Encounter RTL Compiler*

- **Using CPF for Multiple Supply Voltage Designs**
- **Using CPF for Designs Using Power Shutoff Methodology**
- **Using CPF for Designs Using Dynamic Voltage Frequency Scaling**
- **Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology**

Affected by these commands: 

- `apply_power_intent` on page 1018
- `commit_power_intent` on page 1030
report low_power_intent

report_low_power_intent
    [-design design]
    [-isolation_rule_only | -level_shifter_rule_only
     | -power_domain_only | -state_retention_rule_only]
    [-domain domain] [-quality_check]
    [-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.

- **-quality_check**
  Reports possible issues in the power intent file.

- **-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.
Examples

- The following command lists the power domains in the power intent file.
  
  rc:/> report low_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.
  
  rc:/> report low_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.
  
  rc:/> report low_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
  ...
  SW.iso8
  SW.iso9
The following report lists the quality issues of the power intent file.

rc:/> report low_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 precedence will be applicable.
    Domain : SW
    ISO-Rule : 9
========================================

The following command requests a detailed report. The report prints the details of the rules.

rc:/> report low_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
    isolation_target : to
    sink_off_clamp : 0

....
The following report gives the summary of the power intent file.

```plaintext
rc:/> report low_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

See the following chapters in Low Power in Encounter RTL Compiler

- **Using CPF for Multiple Supply Voltage Designs**
- **Using CPF for Designs Using Power Shutoff Methodology**
- **Using CPF for Designs Using Dynamic Voltage Frequency Scaling**

Affected by these commands:  
- `apply_power_intent` on page 1018  
- `commit_power_intent` on page 1030  
- `read_power_intent` on page 1032
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.
**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
```

<table>
<thead>
<tr>
<th>PIN/BUS</th>
<th>DIRECTION</th>
<th>IS_BUS</th>
<th>BUS_TYPE</th>
<th>CAPACITANCE</th>
<th>RELATED_PIN</th>
<th>TIMING_TYPE</th>
<th>TIMING_SENSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>output</td>
<td>YES</td>
<td>RF32X32_DATA</td>
<td>0.002</td>
<td>CLK</td>
<td>rising_edge</td>
<td>non_unate</td>
</tr>
<tr>
<td>D</td>
<td>input</td>
<td>YES</td>
<td>RF32X32_DATA</td>
<td>0.002</td>
<td>CLK</td>
<td>hold_rising</td>
<td></td>
</tr>
<tr>
<td>CEN</td>
<td>input</td>
<td>NO</td>
<td></td>
<td>0.002</td>
<td>CLK</td>
<td>hold_rising</td>
<td></td>
</tr>
<tr>
<td>WEN</td>
<td>input</td>
<td>NO</td>
<td></td>
<td>0.010</td>
<td>CLK</td>
<td>hold_rising</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>input</td>
<td>YES</td>
<td>RF32X32_ADDRESS</td>
<td>0.008</td>
<td>CLK</td>
<td>hold_rising</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLOCK</th>
<th>DIRECTION</th>
<th>CAPACITANCE</th>
<th>MIN_TRANSITION</th>
<th>MIN_PULSE_WIDTH_HIGH</th>
<th>MIN_PULSE_WIDTH_LOW</th>
<th>MIN_PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>input</td>
<td>0.038</td>
<td>1.000</td>
<td>0.055</td>
<td>0.130</td>
<td>0.867</td>
</tr>
</tbody>
</table>
report messages

report messages [-all] [-include_suppressed] [-error]
[-warning] [-info] [> file]

Summarizes the information, warning and error messages that have been issued by RTL Compiler 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, RTL Compiler 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.

Options and Arguments

- all       Reports all messages since you started this RTL Compiler run.
- error     Reports the error messages.
- file      Specifies the name of the file to which to write the report.
- include_suppressed Reports those messages that were suppressed through the suppress_messages command.
- info      Reports the information messages.
- warning   Reports the warning messages.

Examples

Note: The following examples all apply to the same RTL Compiler session.

The following example is the first request to report messages in a session.

cpy:/> report messages

==================
Message Summary
==================

<table>
<thead>
<tr>
<th>Num</th>
<th>Sev</th>
<th>Id</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Info</td>
<td>ELAB-VLOG-9</td>
<td>Variable has no fanout. This variable is not driving anything and will be simplified</td>
</tr>
</tbody>
</table>
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.
  ```
  rc:/> 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.
  ```
  rc:/> 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:
  ```
  rc:/> report messages -all -error
  ```

**Related Information**

**Summarizing Messages** in Using Encounter RTL Compiler
**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`.

```bash
rc:/> report mode i1
------------------------------
Mode Group setup of i1
------------------------------
  Mode Name         Status
slavePushout        ACTIVE
masterCapture       Inactive
```

**Related Information**

Related command: `dc::set_mode`
report multibit_inferencing


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.

- **-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.
  **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.

-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.

Examples

The following command reports the multibit cells in the libraries.

```
rc:/> report multibit_inferencing -lib
```

The following command reports the multibit cells in the libraries.

```
rc:/> report multibit_inferencing -lib
```

<table>
<thead>
<tr>
<th>Comb_Mbit libcell</th>
<th>Avoid</th>
<th>Bitwidth</th>
<th>Leakege Power</th>
<th>Area</th>
<th>Library</th>
<th>Library Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND2_2B</td>
<td>false</td>
<td>2</td>
<td>1.23</td>
<td>1.10</td>
<td>XYZ_multibit</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seq_Mbit libcell</th>
<th>Avoid</th>
<th>Multibit_Usable</th>
<th>Bitwidth</th>
<th>Leakege Power</th>
<th>Area</th>
<th>Library</th>
<th>Library Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFF_1_2B</td>
<td>false</td>
<td>true</td>
<td>2</td>
<td>14.32</td>
<td>7.53</td>
<td>XYZ_multibit</td>
<td>ip_domain</td>
</tr>
<tr>
<td>DFF_1_4B</td>
<td>false</td>
<td>true</td>
<td>4</td>
<td>14.32</td>
<td>15.05</td>
<td>XYZ_multibit</td>
<td>ip_domain</td>
</tr>
<tr>
<td>DFF_1_8B</td>
<td>false</td>
<td>true</td>
<td>8</td>
<td>14.32</td>
<td>30.11</td>
<td>XYZ_multibit</td>
<td>ip_domain</td>
</tr>
<tr>
<td>DFF_1_2B</td>
<td>false</td>
<td>true</td>
<td>2</td>
<td>14.32</td>
<td>7.53</td>
<td>XYZ_multibit</td>
<td>gp_domain</td>
</tr>
<tr>
<td>DFF_1_4B</td>
<td>false</td>
<td>true</td>
<td>4</td>
<td>14.32</td>
<td>15.05</td>
<td>XYZ_multibit</td>
<td>gp_domain</td>
</tr>
<tr>
<td>DFF_1_8B</td>
<td>false</td>
<td>true</td>
<td>8</td>
<td>14.32</td>
<td>30.11</td>
<td>XYZ_multibit</td>
<td>gp_domain</td>
</tr>
</tbody>
</table>
The following command reports the combinational multibit cells used in the design.

```
rc:/> report multibit_inferencing -comb
```

Combinational Multibit cells usage statistics
-----------------------------------------------
Total Combinational instances merged: 2
Total Combinational instances not merged: 0

<table>
<thead>
<tr>
<th>Comb_Mbit libcell</th>
<th>Bitwidth</th>
<th>Count</th>
<th>Total Area</th>
<th>Multibit Conversion %</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND2_2B</td>
<td>2</td>
<td>1</td>
<td>1.10</td>
<td>100.00</td>
<td>XYZ_multibit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td></td>
<td>1.10</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

The following command reports the combinational multibit cells used in the design with their leakage power information.

```
rc:/> report multibit_inferencing -comb -power
```

Combinational Multibit cells usage statistics
-----------------------------------------------
Total Combinational instances merged: 2
Total Combinational instances not merged: 0

<table>
<thead>
<tr>
<th>Comb_Mbit libcell</th>
<th>Bitwidth</th>
<th>Count</th>
<th>Total Area</th>
<th>Total Power(nW)</th>
<th>Multibit Conversion %</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND2_2B</td>
<td>2</td>
<td>1</td>
<td>1.10</td>
<td>1.23</td>
<td>100.00</td>
<td>XYZ_multibit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1</td>
<td></td>
<td>1.10</td>
<td>1.23</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

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.

```
rc:/> report multibit_inferencing -seq
```

Sequential Multibit cells usage statistics
---------------------------------------------

<table>
<thead>
<tr>
<th>Merged</th>
<th>Not Merged</th>
<th>Multibit Conversion %</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Sequentials</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>-FlipFlops</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>-Latches</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Seq_Mbit libcell
-----------------
<table>
<thead>
<tr>
<th>Bitwidth</th>
<th>Count</th>
<th>Total Area</th>
<th>Multibit Conversion %</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFF_1_2B (F)</td>
<td>2</td>
<td>1</td>
<td>7.53</td>
<td>14.29</td>
</tr>
<tr>
<td>DFF_1_4B (F)</td>
<td>4</td>
<td>1</td>
<td>15.05</td>
<td>28.57</td>
</tr>
<tr>
<td>DFF_1_8B (F)</td>
<td>8</td>
<td>1</td>
<td>30.11</td>
<td>57.14</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>52.69</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>
The following command lists the number of instances that did not get merged per reason.

```
rc:// 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 incremental optimization.

Sequential Multibit not merged statistics
===================================================================================================================
Total Sequential instances not merged: 1318
<table>
<thead>
<tr>
<th>Reason of not merging</th>
<th>Sequential Instances</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INST_NO_VALID_REPLACEMENT</td>
<td>1312</td>
<td>No valid replacement / No valid multibit libcell present</td>
</tr>
<tr>
<td>INST_SPLIT</td>
<td>5</td>
<td>Single bit created after splitting the multibit</td>
</tr>
<tr>
<td>INST_WORSE_QOR</td>
<td>1</td>
<td>Move not accepted as it is giving worse QoR</td>
</tr>
</tbody>
</table>

The following command lists the instances that did not get merged with the reason.

```
rc:// 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]
f_reg[3][9]
---------------------------------------------------------------
The following command shows the mapping between any multibit instance to the corresponding original single bit instances.

```
rc:/> 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
```

Related Information

**Mapping to Multibit Cells** in *Encounter RTL Compiler Synthesis Flows*
**report net_cap_calculation**

```text
report net_cap_calculation net [> 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 498
report net_delay_calculation

report net_delay_calculation [-driver_pin {port|pin}...]
  [-load_pin {port|pin}...] [> 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.

 -load_pin {port|pin}
   Specifies the ending port or pin on which to obtain the net delay.

file
   Redirects the report to the specified file.

Example

The following command provides a report based on the in1[0] driver pin.

rc:/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)

<table>
<thead>
<tr>
<th>From pin</th>
<th>To pin</th>
<th>Wire res of net</th>
<th>Wire cap of net of net</th>
<th>Fanout to pin</th>
<th>Pin cap of all to pins</th>
<th>Total pin cap of net to pin delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>in1[0]</td>
<td>instl/g43/A</td>
<td>0.000</td>
<td>7.2</td>
<td>1</td>
<td>5.3</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Related Information

Affected by this attribute: tree_type
**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 496
report nets


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 Specifies an integer value and reports nets whose fanouts are below the given threshold value.

-minfanout 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
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.

rc:\>
report nets -cap_worst 5

The following example sorts the nets in the top level by the number of loads.

rc:\>
report nets -sort load

The following example provides specific information on the n_0 ai net.

rc:\>
report net n_0 ai
The following example reports all the nets in the top level of the current design whose fanout is less than 2.

```
rc:\> report net -maxfanout 2
```

```
================================================================
<table>
<thead>
<tr>
<th>Wire</th>
<th>Wire</th>
<th>Wireload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Loads Drivers Cap(FF) Res(k-ohm) Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in1a[0] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in1a[1] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in1b[0] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in1b[1] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out2a[0] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out2a[1] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out2b[0] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out2b[1] 1 1 0.4 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out3a[0] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out3a[1] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out3b[0] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out3b[1] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out4a[0] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out4a[1] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out4b[0] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>out4b[1] 1 1 0.0 0.000 AL_SMALL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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.

```
rc:/> report net -pin "g22/A bx_reg2/D"
```

```
================================================================
<p>| Total Slew Slew |</p>
<table>
<thead>
<tr>
<th>Net Cap(FF) Rise Fall Driver(s) Load(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_0 20.3 0.0 0.0 g24/Y g23/A g22/A bx_reg3/CK</td>
</tr>
<tr>
<td>ai 12.0 0.0 0.0 ai g25/A bx_reg2/D</td>
</tr>
</tbody>
</table>
```

Related Information

Generating a Net Report in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler
report opcg_equivalents

report opcg_equivalents [-pinVal] [design] [> file]

Reports the OPCG-equivalency mappings specified using the set_opcg_equivalent command(s).

Options and Arguments

\textit{design} \hspace{1cm} Specifies the design for which you want to report the OPCG-equivalency mappings.

\textit{file} \hspace{1cm} Specifies the name of the file to which to write the report.

\textit{-pinVal} \hspace{1cm} 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.

```shell
rc:/> report opcg_equivalents
OPCG equivalent table contains 2 entries {
  {
    Scan cell: /libraries/myspecial/libcells/SDFFQ_X1M
    Opog cell: /libraries/myspecial/libcells/S2DFFQ_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
    Opog cell: /libraries/myspecial/libcells/S2DFFQ_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: \textit{reset_opcg_equivalent} on page 894

set_opcg_equivalent on page 898

Related commands: \textit{replace_opcg_scan} on page 882
**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 (Horizontal, Vertical or Undefined), 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.

  ```
  rc:/> report ple
  
  ==============================================================
  Generated by: Encounter(R) RTL Compiler 10.1.100
  Generated on: Apr 30 2010  03:29:32 pm
  Module: DTMF_CHIP
  Technology libraries:  tsmc18 1.0
                         tps973g 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])

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Direction</th>
<th>Utilization</th>
<th>Capacitance / Length (pF/micron)</th>
<th>Data source:</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>H</td>
<td>1.00</td>
<td>0.000274</td>
<td>cap_table_file</td>
</tr>
<tr>
<td>M2</td>
<td>V</td>
<td>1.00</td>
<td>0.000242</td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>H</td>
<td>1.00</td>
<td>0.000242</td>
<td></td>
</tr>
<tr>
<td>M4</td>
<td>V</td>
<td>1.00</td>
<td>0.000242</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>H</td>
<td>1.00</td>
<td>0.000242</td>
<td></td>
</tr>
</tbody>
</table>
  ```
### Resistance

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Direction</th>
<th>Utilization</th>
<th>Resistance / Length (ohm/micron)</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal1</td>
<td>H</td>
<td>1.00</td>
<td>0.439130</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal2</td>
<td>V</td>
<td>1.00</td>
<td>0.360714</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal3</td>
<td>H</td>
<td>1.00</td>
<td>0.360714</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal4</td>
<td>V</td>
<td>1.00</td>
<td>0.360714</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal5</td>
<td>H</td>
<td>1.00</td>
<td>0.360714</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal6</td>
<td>V</td>
<td>1.00</td>
<td>0.102273</td>
<td>lef_library</td>
</tr>
</tbody>
</table>

### Area

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Direction</th>
<th>Utilization</th>
<th>Area / Length (micron)</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal1</td>
<td>H</td>
<td>1.00</td>
<td>0.230000</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal2</td>
<td>V</td>
<td>1.00</td>
<td>0.280000</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal3</td>
<td>H</td>
<td>1.00</td>
<td>0.280000</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal4</td>
<td>V</td>
<td>1.00</td>
<td>0.280000</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal5</td>
<td>H</td>
<td>1.00</td>
<td>0.280000</td>
<td>lef_library</td>
</tr>
<tr>
<td>Metal6</td>
<td>V</td>
<td>1.00</td>
<td>0.440000</td>
<td>lef_library</td>
</tr>
</tbody>
</table>

**Related Information**

“Checking the Physical Layout Estimation Information” in [Simple PLE Flow, Spatial Flow, and RC-P Flow](/content/designphysically) in *Design with RTL Compiler Physical*
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.

```bash
rc:/> report port -delay -driver -load ck1 e_out[6] ena
```

<table>
<thead>
<tr>
<th>Port</th>
<th>Dir</th>
<th>Clock</th>
<th>Rise Delay</th>
<th>Fall Delay</th>
<th>Ext Delay</th>
<th>Exception Object/Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ck1</td>
<td>in</td>
<td>CLK1</td>
<td>700.0</td>
<td>700.0</td>
<td>in_del_1</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLK2</td>
<td>500.0</td>
<td>500.0</td>
<td>in_del_2</td>
<td></td>
</tr>
<tr>
<td>e_out[6]</td>
<td>out</td>
<td>CLK1</td>
<td>200.0</td>
<td>200.0</td>
<td>ou_del_1</td>
<td>del_1 (path_delay)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLK2</td>
<td>300.0</td>
<td>300.0</td>
<td>ou_del_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>300.0</td>
<td>300.0</td>
<td>outrxt1</td>
<td></td>
</tr>
<tr>
<td>ena</td>
<td>inout</td>
<td>CLK1</td>
<td>700.0</td>
<td>700.0</td>
<td>in_del_1</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLK2</td>
<td>500.0</td>
<td>500.0</td>
<td>in_del_2</td>
<td></td>
</tr>
<tr>
<td>ena</td>
<td>inout</td>
<td>CLK1</td>
<td>200.0</td>
<td>200.0</td>
<td>ou_del_1</td>
<td>del_1 (path_delay)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLK2</td>
<td>300.0</td>
<td>300.0</td>
<td>ou_del_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>300.0</td>
<td>300.0</td>
<td>outrxt1</td>
<td></td>
</tr>
</tbody>
</table>
Related Information

Generating a Port Report in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler
### report power

```shell
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.

- **-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.
  **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.

If you specify neither the `-flat` or `-hier` option, the RC-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.

**-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 \texttt{-rtl_cross_reference} option, this mode is available without any other options or with the \texttt{-verbose} option.

\textbf{file} \hspace{1cm} Sorts by RTL file and line number.

\textbf{Note:} This option applies only to RTL power analysis and is the default for RTL power analysis.

\textbf{internal} \hspace{1cm} Sorts by descending internal power.

\textbf{Note:} If you specified the \texttt{-rtl_cross_reference} option, this mode is only available if you also specified the \texttt{-detail} or \texttt{-verbose} option.

\textbf{leakage} \hspace{1cm} Sorts by descending leakage power.

\textbf{net} \hspace{1cm} Sorts by descending net power.

\textbf{Note:} If you specified the \texttt{-rtl_cross_reference} option, this mode is only available if you also specified the \texttt{-detail} or \texttt{-verbose} option.

\textbf{total} \hspace{1cm} Sorts by descending total power.

\textbf{Note:} If you specified the \texttt{-rtl_cross_reference} option, this mode is available without any other options or with the \texttt{-verbose} option.

The following modes are available for \textit{net}-based reports:

\textbf{dynamic (default)} \hspace{1cm} Sorts by descending total dynamic power.

\textbf{load} \hspace{1cm} Sorts by descending capacitive load on the net.

\textbf{net} \hspace{1cm} Sorts by descending total switching power.

\textbf{prob} \hspace{1cm} Sorts by descending probability of the nets being high.

\textbf{rate} \hspace{1cm} 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.
Examples

- The following command requests a basic RTL power analysis of design `mult_bit_muxed_add`.

  ```bash
  rc:/ > build_rtl_power_models -clean_up_netlist
  rc:/ > report power -rtl_cross_reference
  
  ... Technology library: xxx yy  
  Operating conditions: _nominal_ (balanced_tree)  
  Wireload mode: enclosed
  
  Design Power(nW) Power(nW) Power(nW)
  ---------------------------------------
  mult_bit_muxed_add 71.146 1683.918 1755.064
  
  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.

  ```bash
  rc:/ > report power -rtl -flat
  
  ... Technology library: xxx yy  
  Operating conditions: _nominal_ (balanced_tree)  
  Wireload mode: enclosed
  
  Design Power(nW) Power(nW) Power(nW)
  ---------------------------------------
  mult_bit_muxed_add 71.146 1683.918 1755.064
  
  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
  ```
The following command requests a detailed RTL power analysis report.

```
rc:// report power -rtl -detail -flat
```

```
Module:                mult_bit_muxed_add
```

<table>
<thead>
<tr>
<th>Design</th>
<th>Leakage Power(nW)</th>
<th>Internal Power(nW)</th>
<th>Net Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mult_bit_muxed_add</td>
<td>71.146</td>
<td>1148.102</td>
<td>535.815</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Input</th>
<th>Leakage Power(nW)</th>
<th>Internal Power(nW)</th>
<th>Net Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[1]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>a[0]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>b[1]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>b[0]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>c[1]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>c[0]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>d[1]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>d[0]</td>
<td>0.000</td>
<td>0.000</td>
<td>34.050</td>
</tr>
<tr>
<td>s</td>
<td>0.000</td>
<td>0.000</td>
<td>52.536</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File</th>
<th>Row</th>
<th>RTL Line Instances</th>
<th>Leakage Power(nW)</th>
<th>Internal Power(nW)</th>
<th>Net Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>muxed_add.v</td>
<td>8</td>
<td>{if (s) begin} mux..8_5_g1 mux_y_8_5_g1</td>
<td>14.199</td>
<td>469.091</td>
<td>82.695</td>
</tr>
<tr>
<td>muxed_add.v</td>
<td>9</td>
<td>{ y = a + c ; } g2 g2</td>
<td>28.473</td>
<td>334.117</td>
<td>62.021</td>
</tr>
<tr>
<td>muxed_add.v</td>
<td>12</td>
<td>{ y = b + d ; } g2 g2</td>
<td>28.473</td>
<td>344.894</td>
<td>66.156</td>
</tr>
</tbody>
</table>
The following command shows the clock tree power estimation for clock \texttt{clk}.

\begin{verbatim}
rc:/> report power -clock_tree iCLK1 -width 5 -height 5
============================================================
Generated by: Encounter(r) RTL Compiler 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
\end{verbatim}

The following command reports the power (after synthesis) at the current level in the hierarchy, which is the design. With the \texttt{-hier} option specified, the report lists the design and its hierarchical instances.

\begin{verbatim}
rc:/designs/mult_bit_muxed_add> report power -hier
============================================================
...  
============================================================
<table>
<thead>
<tr>
<th>Instance</th>
<th>Cells</th>
<th>Leakage Power(nW)</th>
<th>Dynamic Power(nW)</th>
<th>Total Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mult_bit_muxed_add</td>
<td>6</td>
<td>71.146</td>
<td>1683.862</td>
<td>1755.008</td>
</tr>
<tr>
<td>ma0</td>
<td>3</td>
<td>35.573</td>
<td>724.869</td>
<td>760.442</td>
</tr>
<tr>
<td>ma1</td>
<td>3</td>
<td>35.573</td>
<td>643.092</td>
<td>678.665</td>
</tr>
</tbody>
</table>
\end{verbatim}
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.

```
rc:/designs/mult_bit_muxed_add> report power -verbose
```

```
Leakage Internal Net Dynamic Total
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.

```
rc:/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.

```
rc:/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
```
The following command reports the power for an instance and sorts the report according to descending net power.

```bash
rc:/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.

```bash
rc:/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
```

<table>
<thead>
<tr>
<th>Net</th>
<th>(asserted *)</th>
<th>Power (nW)</th>
<th>Prob. Rate (/ns)</th>
<th>Cap. (nF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[1]</td>
<td>27.945</td>
<td>0.500</td>
<td>0.021</td>
<td>2.300</td>
</tr>
</tbody>
</table>
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.

```
rc:/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

<table>
<thead>
<tr>
<th>Instance</th>
<th>Domain (Voltage)</th>
<th>Cells</th>
<th>Leakage Power(nW)</th>
<th>Dynamic Power(nW)</th>
<th>Total Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>PDcore(0.81v)</td>
<td>142</td>
<td>7.076</td>
<td>9192.022</td>
<td>9199.099</td>
</tr>
<tr>
<td>monitor_power</td>
<td>PDmon(0.81v)</td>
<td>84</td>
<td>3.720</td>
<td>4050.236</td>
<td>4053.956</td>
</tr>
<tr>
<td>adder_counter</td>
<td>PAdd(0.81v)</td>
<td>25</td>
<td>1.215</td>
<td>1573.961</td>
<td>1575.175</td>
</tr>
<tr>
<td>bcd_counter</td>
<td>PDbcd(0.81v)</td>
<td>12</td>
<td>1.088</td>
<td>1697.572</td>
<td>1698.659</td>
</tr>
<tr>
<td>binary_counter</td>
<td>PDbin(0.81v)</td>
<td>18</td>
<td>1.012</td>
<td>1565.877</td>
<td>1566.888</td>
</tr>
</tbody>
</table>
```

```
rc:/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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Domain Cells</th>
<th>Leakage Power(nW)</th>
<th>Dynamic Power(nW)</th>
<th>Total Power(nW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>counter</td>
<td>umc_1v</td>
<td>142</td>
<td>11.023</td>
<td>33056.713</td>
</tr>
<tr>
<td>monitor_power</td>
<td>umc_1v</td>
<td>84</td>
<td>5.805</td>
<td>14774.642</td>
</tr>
<tr>
<td>adder_counter</td>
<td>umc_1v</td>
<td>25</td>
<td>1.884</td>
<td>4185.202</td>
</tr>
<tr>
<td>bcd_counter</td>
<td>umc_1v</td>
<td>12</td>
<td>1.697</td>
<td>6138.779</td>
</tr>
<tr>
<td>binary_counter</td>
<td>umc_1v</td>
<td>18</td>
<td>1.569</td>
<td>6542.145</td>
</tr>
</tbody>
</table>
```
Related Information

See the following sections in *Low Power in Encounter RTL Compiler*

- RTL Power Analysis
- Reporting Clock Tree Power
- Reporting on All Power Components
- Reporting Leakage Power
- Reporting Dynamic Power

Affected by these commands:  
  - `build_rtl_power_models` on page 973
  - `synthesize` on page 377

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`
**report power_domain**

```
report power_domain
    [-detail] [-power_mode] [-qor] [> file]
```

Reports power domain related information.

**Note:** An asterisk (*) identifies the default power domain.

Without any options specified, a summary report is given which shows for each power domain

- The name of the shutoff signal
- Whether the power domain is an always-on domain
- Whether the power domain is externally controlled
- The operating voltage in the default power mode

**Options and Arguments**

- **-detail** Prints the summary information and the information you would get by specifying the `-power_mode` and `-qor` options.

- **file** Specifies the name of the file to which the report is to be written.

- **-power_mode** Prints the operating voltage of each power domain in each power mode.

- **-qor** Prints the following information for each power domain:
  - The cell area
  - The percentage of nets with user-asserted switching activities
  - The dynamic power consumption
  - The leakage power consumption

**Note:** The results are given for the current power mode and are affected by the setting of the `lp_power_unit` attribute.
Examples

- The following example shows the basic report (with no options specified).

  ```
  rc:/designs/top> report power_domain
  Summary
  =========
  Name Shut-off signal Always on External controlled Voltage(V)
  LD1 true false 1.2
  PD1(*) false false 0.8
  PD2 pm_inst/pse_enable[0] false false 0.8
  PD3 pm_inst/pse_enable[1] false false 0.8
  PD4 pm_inst/pse_enable[2] false false 0.8
  ```

- The following example shows the power mode information. The default power mode is marked with an asterisk.

  ```
  rc:/designs/top> report power_domain -power_mode
  Power Modes
  =============
  Power Domain PM1 PM2 PM3 PM4
  LD1 1.2 1.2 1.2 1.2
  PD1(*) 0.8 0.8 0.8 0.8
  PD2 0.8 OFF OFF OFF
  PD3 0.8 OFF OFF
  PD4 0.8 OFF
  ```

Related Information

Affected by these commands: read_power_intent on page 1032
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.

```plaintext
rc:\> report proto -feasible_targets

============================================================
...
============================================================

<table>
<thead>
<tr>
<th>No.</th>
<th>Endpoint</th>
<th>Path Adjust Value(ps)</th>
<th>Path Adjust Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>top/piano/u_reg/d</td>
<td>1500</td>
<td>proto_ft_adj_1</td>
</tr>
<tr>
<td>2</td>
<td>top/piano/p_reg/d</td>
<td>1500</td>
<td>proto_ft_adj_2</td>
</tr>
<tr>
<td>3</td>
<td>top/piano/p_reg/d</td>
<td>1500</td>
<td>proto_ft_adj_2</td>
</tr>
<tr>
<td>4</td>
<td>top/q_reg1/d</td>
<td>450</td>
<td>proto_ft_zipped_1</td>
</tr>
<tr>
<td>5</td>
<td>top/q_reg2/d</td>
<td>450</td>
<td>proto_ft_zipped_2</td>
</tr>
</tbody>
</table>
```

The following command prints a list of incomplete RTL scenarios.

```
rc:/> 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

Express Flow in Encounter RTL Compiler Synthesis Flows

Affected by these commands: elaborate

synthesize

Related attributes: proto_feasible_target
proto_hdl
Report qor

```
report qor
   [-levels_of_logic] [-nopower | -power]
   [-exclude_constant_nets]
   [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 Encounter® 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.
- **-nopower**: Prevents power computation for this report.
- **-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.

rc:/> 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
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.

rc:
> 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 (DFF_sdo_1)
Average Fanout 2.2
Terms to net ratio 2.6
Terms to instance ratio 3.6
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 to the table under Timing. The rest of the report does not change.

rc:\> report qor -levels_of_logic

```
============================================================
...  
============================================================
```

Timing
------

<table>
<thead>
<tr>
<th>Cost Group</th>
<th>Critical Path Slack TNS</th>
<th>No of gates on Critical Path</th>
<th>Violating Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2C</td>
<td>No paths</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2O</td>
<td>2015.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CLK1</td>
<td>2019.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>No paths</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>I2C</td>
<td>1707.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I2O</td>
<td>No paths</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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
------

```
<table>
<thead>
<tr>
<th>Mode</th>
<th>Cost Group</th>
<th>Critical Path Slack</th>
<th>Violating Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>default</td>
<td>No paths</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>I2C</td>
<td>-202.6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C2O</td>
<td>-116.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C2C</td>
<td>No paths</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I2O</td>
<td>No paths</td>
<td></td>
</tr>
<tr>
<td>m2</td>
<td>default</td>
<td>-202.6</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>I2C</td>
<td>No paths</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C2O</td>
<td>No paths</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C2C</td>
<td>No paths</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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.

```
rc:/> report qor
============================================================
Generated by: Encounter(R) RTL Compiler version ...
Interconnect mode: placement
Area mode: physical library
============================================================
Timing
-------
Clock Period
----------
vclk01  5000.0
vclk02  6000.0
....

<table>
<thead>
<tr>
<th>Cost Group</th>
<th>Critical Path Slack</th>
<th>TNS</th>
<th>Violating Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>No paths</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vclk1</td>
<td>-4944.1</td>
<td>-12686</td>
<td>38</td>
</tr>
<tr>
<td>vclk01</td>
<td>No paths</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>-12686</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>

Instance Count
------------
Leaf Instance Count 5977
Sequential Instance Count 546
Combinational Instance Count 5431
Hierarchical Instance Count 26

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.

rc:/ > 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

Related Information

Affected by these attributes: max_dynamic_power
max_leakage_power
**report scan_compressibility**

*report scan_compressibility -directory atpg_directory [> file]*

Reports the scan compressibility of a design.

**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`.

  `rc:>report scan_compressibility -directory asc0`

Analyzing from work directory ’asco’ .....  

------------------------------------------------------------------------------------------------------

<table>
<thead>
<tr>
<th>IC</th>
<th>TATR</th>
<th>TDVR</th>
<th>ATCov.</th>
<th>CL</th>
<th>Pat-comp</th>
<th>Pat-fs</th>
<th>Cycles</th>
<th>Runtime</th>
<th>Gatecount</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>1.0</td>
<td>1.0</td>
<td>100.00%</td>
<td>63</td>
<td>-</td>
<td>3</td>
<td>336</td>
<td>00:00:00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>0.5</td>
<td>99.91%</td>
<td>32</td>
<td>9</td>
<td>5</td>
<td>1258</td>
<td>00:10:01</td>
<td>437</td>
<td>24774.3999999</td>
</tr>
</tbody>
</table>

Achieved compression table without fullscan topup vectors

------------------------------------------------------------------------------------------------------

<table>
<thead>
<tr>
<th>IC</th>
<th>TATR</th>
<th>TDVR</th>
<th>ATCov.</th>
<th>CL</th>
<th>Pat-comp</th>
<th>Pat-fs</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>1.0</td>
<td>1.0</td>
<td>100.00%</td>
<td>63</td>
<td>-</td>
<td>-</td>
<td>336</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>1.2</td>
<td>98.76%</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>838</td>
</tr>
</tbody>
</table>

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  
Pat-fs - No. of fullscan test patterns  
Runtime - Atpg runtime  
fs - fullscan run
Related Information

Analyzing and Reporting Scan Compressibility in Design for Test in Encounter RTL Compiler.

Affected by this command: analyze_scan_compressibility on page 633
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.

rc:/> report sequential

```
-------------------------------------------------------------
Generated by:  Version
Generated on:  Date
Module:  test
Technology library:  slow 1.5
Operating conditions:  slow (balanced_tree)
Wireload mode:  segmented
-------------------------------------------------------------

<table>
<thead>
<tr>
<th>Register</th>
<th>File</th>
<th>Row</th>
<th>Column</th>
<th>Instantiated/Inferred</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>sync_rst_reg</td>
<td>all.v</td>
<td>12</td>
<td>16</td>
<td>inferred</td>
<td>flip-flop synchronous reset</td>
</tr>
<tr>
<td>async_rst_reg</td>
<td>all.v</td>
<td>21</td>
<td>27</td>
<td>inferred</td>
<td>flip-flop asynchronous reset</td>
</tr>
<tr>
<td>sync_set_reg</td>
<td>all.v</td>
<td>30</td>
<td>37</td>
<td>inferred</td>
<td>flip-flop asynchronous set</td>
</tr>
<tr>
<td>no_rst_reg</td>
<td>all.v</td>
<td>39</td>
<td>45</td>
<td>inferred</td>
<td>flip-flop</td>
</tr>
<tr>
<td>sync_preset_reg</td>
<td>all.v</td>
<td>44</td>
<td>58</td>
<td>inferred</td>
<td>flip-flop synchronous set</td>
</tr>
<tr>
<td>q_reg</td>
<td>all.v</td>
<td>6</td>
<td>12</td>
<td>inferred</td>
<td>latch</td>
</tr>
</tbody>
</table>
```

■ The following example reports all the sequential elements in the design.

rc:/> report sequential -hier

```
report sequential: prints a sequential instance report.

```

```
<table>
<thead>
<tr>
<th>Register</th>
<th>File</th>
<th>Row</th>
<th>Column</th>
<th>Instantiated/Inferred</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2/m3/m4/m5/o_m5_0_reg_1</td>
<td>hier.v</td>
<td>25</td>
<td>22</td>
<td>instantiated</td>
<td>flip-flop synchronous enable</td>
</tr>
<tr>
<td>m2/m3/m4/m5/o_m5_0_reg_2</td>
<td>hier.v</td>
<td>27</td>
<td>22</td>
<td>instantiated</td>
<td>flip-flop synchronous enable</td>
</tr>
<tr>
<td>m2/m3/m4/m5/o_m5_1_reg_0</td>
<td>hier.v</td>
<td>30</td>
<td>22</td>
<td>instantiated</td>
<td>flip-flop synchronous enable</td>
</tr>
<tr>
<td>m2/m3/m4/m5/o_m5_0_reg_0</td>
<td>hier.v</td>
<td>23</td>
<td>22</td>
<td>instantiated</td>
<td>flip-flop synchronous enable</td>
</tr>
<tr>
<td>m2/o_m2_clk1_0_reg_2</td>
<td>hier.v</td>
<td>174</td>
<td>27</td>
<td>instantiated</td>
<td>flip-flop synchronous enable</td>
</tr>
</tbody>
</table>
```

```
The following example reports a timing model.

```
rc:/> report sequential
=============================================================================
...                                                                                   
=============================================================================
```

<table>
<thead>
<tr>
<th>Register</th>
<th>File</th>
<th>Row</th>
<th>Column</th>
<th>Instantiated/Inferred</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>clockgate/g2clatch_tlat timing_model.v</td>
<td>22</td>
<td>29</td>
<td>instantiated</td>
<td>timing_model</td>
<td></td>
</tr>
</tbody>
</table>

The following example reports the sequential elements that were removed during optimization.

```
rc:/> report sequential -deleted_seqs
=============================================================================
...                                                                                   
=============================================================================
```

<table>
<thead>
<tr>
<th>Reason</th>
<th>Instance Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>merged</td>
<td>ctrl_wr_reg[36] merged with ctrl_wr_reg[0]</td>
</tr>
<tr>
<td>merged</td>
<td>ctrl_wr_reg[40] merged with ctrl_wr_reg[12]</td>
</tr>
<tr>
<td>merged</td>
<td>ctrl_wr_reg[46] merged with ctrl_wr_reg[3]</td>
</tr>
<tr>
<td>merged</td>
<td>ctrl_wr_reg[47] merged with ctrl_wr_reg[10]</td>
</tr>
<tr>
<td>merged</td>
<td>ctrl_wr_reg[41] merged with ctrl_wr_reg[13]</td>
</tr>
<tr>
<td>unloaded</td>
<td>ctrl_wr_reg[32]</td>
</tr>
<tr>
<td>unloaded</td>
<td>ctrl_wr_reg[20]</td>
</tr>
<tr>
<td>unloaded</td>
<td>ctrl_wr_reg[28]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

If the delete_unloaded_seqs root attribute is set to false, the report may look like:

```
rc:/> report sequential -deleted_seqs
=============================================================================
...                                                                                   
=============================================================================
```

<table>
<thead>
<tr>
<th>Reason</th>
<th>Instance Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>merged</td>
<td>( * ) ctrl_wr_reg[36] merged with ctrl_wr_reg[0]</td>
</tr>
<tr>
<td>merged</td>
<td>( * ) ctrl_wr_reg[40] merged with ctrl_wr_reg[12]</td>
</tr>
<tr>
<td>merged</td>
<td>( * ) ctrl_wr_reg[46] merged with ctrl_wr_reg[3]</td>
</tr>
<tr>
<td>merged</td>
<td>( * ) ctrl_wr_reg[47] merged with ctrl_wr_reg[10]</td>
</tr>
<tr>
<td>merged</td>
<td>( * ) ctrl_wr_reg[41] merged with ctrl_wr_reg[13]</td>
</tr>
<tr>
<td>constant</td>
<td>( * ) ctrl_wr_reg[52]</td>
</tr>
<tr>
<td>constant</td>
<td>( * ) ctrl_wr_reg[53]</td>
</tr>
</tbody>
</table>

( * ) indicates that the instance is optimized but not deleted because root attribute 'delete_unloaded_seq' is set to false.
report slew_calculation

report slew_calculation  pin [-rise | -fall] [> 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.

pin
Specifies the cell driver pin.
report summary

report summary
    [-mode mode] [-all]
    [design]... [> 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
    design

Reports all timing and DRC violations in the design.

    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.

    -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.

rc:/> report summary

===================================================================================================
Generated by: RTL Compiler (RC) 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

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

See the following sections in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler

- Generating a Summary Report
- Performing Multi-Mode Timing Analysis

Affected by this command: synthesize on page 377
create_mode on page 315
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]
    [-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).

Note: For more information on the exact product requirements needed to use this command, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.
-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.
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%.

  ```
  rc:/> report test_power -clock 20
  Computing the test power with the following toggle frequencies:
  ... set clocks @ 20 MHz
  ... set flops @ 50%
  ```

- 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).

  ```
  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 \
-lib $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 \
-lib $simLibs
```

The following command requests to compute the test power from the specified TCF file.

```
report test_power -tcf top.tcf
```

**Related Information**

*Analyzing the Test Power in Design for Test in Encounter RTL Compiler*

Affected by this constraint:  

```
define_dft test_clock
```

on page 762
report timing

report timing [-endpoints] [-summary] [-lint]
  [-full_pin_names] [-physical] [-num_paths integer]
  [-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]
  [-gui [-id integer] | -gui_phys] [> 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.

**Note:** All values are always expressed in picoseconds. This unit cannot be changed.

**Important**

When RTL Compiler 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, RTL Compiler 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.

Where applicable, special footnotes are used as shown in the table below to enhance the usability of the report.

<table>
<thead>
<tr>
<th>Footnote</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(*)</td>
<td>Zero Slack Borrow Limit = Maximum amount that can be borrowed without violating timing on the lending side</td>
</tr>
<tr>
<td>(@)</td>
<td>Annotated capacitance</td>
</tr>
<tr>
<td>(#)</td>
<td>NDR net</td>
</tr>
<tr>
<td>(a)</td>
<td>Net has asynchronous load pins which are being considered ideal</td>
</tr>
<tr>
<td>(b)</td>
<td>Timing paths are broken</td>
</tr>
<tr>
<td>(B)</td>
<td>Net has high fanout (fanout &gt;= 1000)</td>
</tr>
<tr>
<td>(C)</td>
<td>Cell belongs to congested region (applies only to physical flow)</td>
</tr>
<tr>
<td>(e)</td>
<td>Net has extra wire length</td>
</tr>
<tr>
<td>(H)</td>
<td>Estimated wire delay based on physical-aware mapping using virtual buffering as needed.</td>
</tr>
<tr>
<td>(i)</td>
<td>Net is ideal</td>
</tr>
<tr>
<td>(m)</td>
<td>Attribute cell_delay_multiplier is modified for this library cell</td>
</tr>
<tr>
<td>Footnote</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>(P)</td>
<td>Instance is preserved</td>
</tr>
<tr>
<td>(p)</td>
<td>Instance is preserved but may be resized</td>
</tr>
<tr>
<td>(u)</td>
<td>Net has unmapped pin(s)</td>
</tr>
<tr>
<td>(V)</td>
<td>Net with virtual buffer(s)</td>
</tr>
</tbody>
</table>

### Options and Arguments

- **-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 Encounter Digital Implementation System and Encounter Timing System. 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**
  - 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.

- **-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.

- **-gtd**
  - Generates the MRTRF (Machine readable format) output out of RTL Compiler. This is a file format used for representing timing information that can be understood by the Global Timing Debug (GTD) viewer in Encounter. The GTD viewer in Encounter parses this file and represents the timing information in graphical format. This helps illustrate the total slack, failing paths, components contributing the most delay in each path, the total number of violations, and more in the design.
-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.

**Note:** This option has only effect when you run the tool in GUI mode.

-gui_phys

Opens a new timing report dialog and a new Physical Viewer that are linked.

-id integer

In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.

**Note:** This option must be specified with the -gui option.

-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.

-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

**Note:** When combined with the -endpoints option, the number of endpoints is limited to the specified number.

-paths string

Reports only the specified timing restricted paths. Create the string argument using 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.

-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 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.

-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.

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 command reports only the worst path being launched by clock clk1.
  rc:/> report timing -paths [specify_paths -from clk1]

- The following example reports the slack at the four most-constrained endpoints.
  rc:/designs/alu> report timing -endpoints -num_paths 4

- The following example reports the path from input port accum[1] that has the least slack.
  rc:/> report timing -from [find / -port accum[1]]
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.

```plaintext
rc:/designs/alu> report timing -num_paths 4

============================================================================
...                                                                ITHER
============================================================================
path 1:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Fanout</th>
<th>Load</th>
<th>Slew</th>
<th>Delay</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
============================================================================

(clock clock) capture 3500 R
decisions -500 3000

Exception : 'path_adjusts/adj_1' path adjust -500ps
Cost Group : 'I2C_cg' (path_group 'I2C_pg')
Timing slack : 0ps
Start-point : accum[1]
End-point : aluout_reg[7]/D

path 2:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Fanout</th>
<th>Load</th>
<th>Slew</th>
<th>Delay</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
============================================================================

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:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Fanout</th>
<th>Load</th>
<th>Slew</th>
<th>Delay</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
============================================================================

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:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Fanout</th>
<th>Load</th>
<th>Slew</th>
<th>Delay</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
============================================================================

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
```
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.

```
rc:/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
```
The following command reports the path with the least slack that uses a multi_cycle exception named mc_2.

```
rc:/> report timing -exceptions [find / -exception mc_2]
```

The following example reports only a condensed version of the timing report.

```
rc:/> report timing -summary
```

The following example illustrate how to use the -gtd option.

```
rc:/> report_timing -from ..... -to ..... -gtd > viol.mtarpt
Open viol.mtarpt in the global timing debug viewer in Encounter.
```

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.

```
rc:/> report timing -user_derate
Warning : Possible timing problems have been detected in this design. [TIM-11] : The design is ‘top’.
```

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Fanout</th>
<th>Load</th>
<th>Slew</th>
<th>User Delay</th>
<th>Arrival</th>
<th>Derate</th>
<th>(F)</th>
<th>(ps)</th>
<th>(ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(clock clk) launch</td>
<td>0 R</td>
<td>0 R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst1/CK 0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst1/Q fflopd (m) 1 20.4 16 3.000 +443 443 F</td>
<td>443</td>
<td>443</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst2/A +0</td>
<td>443</td>
<td>443</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst2/Y invl 1 25.4 17 1.000 +85 528 R</td>
<td>528</td>
<td>528</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst6/A +0</td>
<td>528</td>
<td>528</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst6/Y buf1 1 20.4 16 1.000 +134 662 R</td>
<td>662</td>
<td>662</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst3/D fflopd_ckn 1.000 +0 662</td>
<td>662</td>
<td>662</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst3/CKN setup 0</td>
<td>762</td>
<td>762</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
The following example shows the timing report when you use the `-lint` option. In this example 5 possible timing problems were found.

```
rc:/> 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. ......
/designed/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. ......
/designed/test/ports_in/in[0]
/designed/test/ports_in/in[1]
/designed/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. ......
/designed/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
- 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
- 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
The following example reports the physical information (if the design has been placed):

```plaintext
rc:/> report timing -physical

==============================================================================
Pin Type Fanout Load Slew Delay Arrival Location (x, y)
(loc) (ps) (ps) (ps) (x, y)
==============================================================================
(clock clock1) launch 0 R
wr_addr_reg[0]/CK 0 118 +0 311 311 R (107640, 51660)
g154/A 118 +0 311
g154/Y (©) CLKMX2X2 105 +206 517 R (102580, 51660)
g146/B 105 +0 517
g146/CO (©) ADDHXL 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.

```plaintext
rc:/> report timing -encounter

==============================================================================
Pin Edge Cell Fanout Load Slew Delay Arrival Required
(loc) (ps) (ps) (ps) (x, y)
==============================================================================
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 example reports a specific path in the sub-bin `1ns_slack_2ns` in parent bin `worst1000_c2`.

```
rc:/> 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.

```
rc:/> 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`.

```
rc:/> 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.

```
rc:/> report timing -timing_path worst1000_c2c/1ns_slack_2ns/p_27 –gui –id 2
```

**Related Information**

See the following sections in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*:

- Checking the Constraints Using the report timing-lint Command
- Performing Multi-Mode Timing Analysis

**Affected by these commands:**
- `create_mode` on page 315
- `specify_paths` on page 351
- `synthesize` on page 377
- `dc::set_timing_derate`

**Affected by this attribute:**
- `cell_delay_multiplier`
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.

    rc:/> 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
**report utilization**

`report utilization [>file]`

Reports the floorplan utilization for the design.

**Options and Arguments**

`file` Redirects the report to the specified file.

**Example**

```
rc:/> report utilization
====================================================================================================
  Generated by:    Encounter(R) RTL Compiler 11.20
  Generated on:    Mar 09 2012  06:11:47 pm
  Module:          fifo
  Technology libraries: slow 1.3
                  physical_cells
  Operating conditions:  slow
  Interconnect mode:  global
  Area mode:          physical library
====================================================================================================
Total cell area (TCA) = 4399.660800000005
Total fixed cell area (TFA) = 0.0
Available row area (ARA) = 8614.305
utilization  { (TCA - TFA)/ARA } = 0.51
```
**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.

```
rc:/> report yield
```

<table>
<thead>
<tr>
<th>Instance</th>
<th>Cells</th>
<th>Cell Area</th>
<th>Cost</th>
<th>Yield %</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu</td>
<td>470</td>
<td>659</td>
<td>1.600606e-05</td>
<td>99.9984</td>
</tr>
<tr>
<td>alul</td>
<td>248</td>
<td>283</td>
<td>7.606708e-06</td>
<td>99.9992</td>
</tr>
<tr>
<td>pcount1</td>
<td>65</td>
<td>92</td>
<td>2.215669e-06</td>
<td>99.9998</td>
</tr>
<tr>
<td>ireg1</td>
<td>33</td>
<td>88</td>
<td>1.629471e-06</td>
<td>99.9998</td>
</tr>
<tr>
<td>accum1</td>
<td>33</td>
<td>88</td>
<td>1.629471e-06</td>
<td>99.9998</td>
</tr>
<tr>
<td>decode1</td>
<td>50</td>
<td>67</td>
<td>1.568901e-06</td>
<td>99.9998</td>
</tr>
</tbody>
</table>

**Related Information**

**Design For Manufacturing Flow** in *Encounter RTL Compiler Synthesis Flows*

- Affected by this command: `read_dfm`
- Related command: `report gates` -yield
- Affected by this attribute: `optimize_yield`
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 RTL Compiler.
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

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Related commands:  statistics add_metric on page 557
                   statistics log on page 558
                   statistics read on page 560
                   statistics remove_metric on page 561
                   statistics report on page 562
                   statistics reset on page 565
statistics run\_stage\_ids on page 566

statistics write on page 567
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

Related Information

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Affects this command: statistics report on page 562
Related command: statistics remove_metric on page 561
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
Related Information

Tracking and Saving QoR Metrics in *Using Encounter RTL Compiler*

Affects these commands: statistics report on page 562
statistics run stage ids on page 566

Related command: statistics run stage ids on page 566
statistics read

Statistics read
   -file file -reset

Reads the statistics information (QoR metrics) from the specified file into RTL Compiler.

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

rc:/> 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

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Affects this command: statistics run_stage_ids on page 566
Related command: statistics write on page 567
Related attributes: statistics run_id
                   statistics run_description
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

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Affects this command: statistics report on page 562
Related command: statistics add_metric on page 557
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

-exclude run_id Specifies the run that you want to compare to the run specified with the -run_id option.

-exclude_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.

```plaintext
statistics report -run_id sample1 -stage_id incremental
```

QOR statistics summary

<table>
<thead>
<tr>
<th>Metric</th>
<th>incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNS.I2O</td>
<td>no_value</td>
</tr>
<tr>
<td>WNS.I2C</td>
<td>1387.9</td>
</tr>
<tr>
<td>WNS.CLK1</td>
<td>2219.9</td>
</tr>
<tr>
<td>WNS.C2C</td>
<td>no_value</td>
</tr>
<tr>
<td>WNS.C2O</td>
<td>2187.2</td>
</tr>
<tr>
<td>WNS.default</td>
<td>no_value</td>
</tr>
<tr>
<td>TNS</td>
<td>0</td>
</tr>
<tr>
<td>Violating_paths</td>
<td>0</td>
</tr>
<tr>
<td>runtime</td>
<td>0.14</td>
</tr>
<tr>
<td>memory</td>
<td>0.00</td>
</tr>
<tr>
<td>Leakage_power</td>
<td>327.19</td>
</tr>
<tr>
<td>Net_power</td>
<td>362656.25</td>
</tr>
<tr>
<td>Internal_power</td>
<td>19369.41</td>
</tr>
<tr>
<td>Clock_gating_instances</td>
<td>0</td>
</tr>
<tr>
<td>total_net_length</td>
<td>n/a</td>
</tr>
<tr>
<td>average_net_length</td>
<td>n/a</td>
</tr>
<tr>
<td>routing_congestion</td>
<td>n/a</td>
</tr>
<tr>
<td>utilization</td>
<td>0.0</td>
</tr>
<tr>
<td>Inverter_count</td>
<td>10</td>
</tr>
<tr>
<td>Buffer_count</td>
<td>0</td>
</tr>
<tr>
<td>timing_model_count</td>
<td>0</td>
</tr>
<tr>
<td>sequential_count</td>
<td>16</td>
</tr>
<tr>
<td>unresolved_count</td>
<td>0</td>
</tr>
<tr>
<td>logic_count</td>
<td>13</td>
</tr>
<tr>
<td>Total_area</td>
<td>538.01</td>
</tr>
<tr>
<td>Cell_area</td>
<td>538.01</td>
</tr>
<tr>
<td>Net_area</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- The following report includes header and footer information defined with the `statistics add_metric -header` and `-footer` options.

QOR statistics summary

```plaintext
Machine_Info rcae010 Intel(R) Xeon(TM) CPU 2.80GHz 2793.238Mhz 32bits 2_cores
Date Aug17-21:56:48
```

<table>
<thead>
<tr>
<th>Metric</th>
<th>elaborate generic global_map incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNS.I2O</td>
<td>n/a no_value no_value no_value</td>
</tr>
<tr>
<td>WNS.I2C</td>
<td>n/a 1533.5 1912.6 1387.9</td>
</tr>
<tr>
<td>WNS.CLK1</td>
<td>n/a no_value 2219.9 2219.9</td>
</tr>
<tr>
<td>WNS.C2C</td>
<td>n/a no_value no_value no_value</td>
</tr>
<tr>
<td>WNS.C2O</td>
<td>n/a 2187.2 2314.1 2187.2</td>
</tr>
<tr>
<td>WNS.default</td>
<td>no_value no_value no_value no_value</td>
</tr>
<tr>
<td>TNS</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>Violating_paths</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

**Elapsed_Time 266**

**Super_Thread_Total_Runtime 245.000**
The following command does a comparison between two runs.

```
statistics report -run_id medium_effort -compare high_effort -stage global_map
QOR statistics summary
```

<table>
<thead>
<tr>
<th>Metric</th>
<th>global_map.medium_effort</th>
<th>global_map.high_effort</th>
<th>%diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNS.I2O</td>
<td>no_value</td>
<td>no_value</td>
<td>n/a</td>
</tr>
<tr>
<td>WNS.I2C</td>
<td>1912.6</td>
<td>1887.6</td>
<td>1.31</td>
</tr>
<tr>
<td>WNS.CLK1</td>
<td>2219.9</td>
<td>2219.9</td>
<td>0.0</td>
</tr>
<tr>
<td>WNS.C2C</td>
<td>no_value</td>
<td>no_value</td>
<td>n/a</td>
</tr>
<tr>
<td>WNS.C2O</td>
<td>2314.1</td>
<td>2314.1</td>
<td>0.0</td>
</tr>
<tr>
<td>WNS.default</td>
<td>no_value</td>
<td>no_value</td>
<td>n/a</td>
</tr>
<tr>
<td>TNS</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>Violating_paths</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>runtime</td>
<td>1.74</td>
<td>0.32</td>
<td>81.61</td>
</tr>
<tr>
<td>memory</td>
<td>8.05</td>
<td>1.11</td>
<td>86.21</td>
</tr>
<tr>
<td>Leakage_power</td>
<td>1730.88</td>
<td>1730.88</td>
<td>0.0</td>
</tr>
<tr>
<td>Net_power</td>
<td>453828.12</td>
<td>453828.12</td>
<td>0.0</td>
</tr>
<tr>
<td>Internal_power</td>
<td>59389.35</td>
<td>59389.35</td>
<td>0.0</td>
</tr>
<tr>
<td>Clock_gating_instances</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>total_net_length</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>average_net_length</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>routing_congestion</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>utilization</td>
<td>0.0</td>
<td>0.0</td>
<td>n/a</td>
</tr>
<tr>
<td>Inverter_count</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>Buffer_count</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>timing_model_count</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>sequential_count</td>
<td>16</td>
<td>16</td>
<td>0.0</td>
</tr>
<tr>
<td>unresolved_count</td>
<td>0</td>
<td>0</td>
<td>n/a</td>
</tr>
<tr>
<td>logic_countT</td>
<td>13</td>
<td>13</td>
<td>0.0</td>
</tr>
<tr>
<td>Total_area</td>
<td>995.70</td>
<td>995.70</td>
<td>0.0</td>
</tr>
<tr>
<td>Cell_area</td>
<td>995.70</td>
<td>995.70</td>
<td>0.0</td>
</tr>
<tr>
<td>Net_area</td>
<td>0.00</td>
<td>0.00</td>
<td>n/a</td>
</tr>
<tr>
<td>state</td>
<td>mapped</td>
<td>mapped</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Related Information

Tracking and Saving QoR Metrics in *Using Encounter RTL Compiler*

Affected by these commands:
- statistics log on page 558
- statistics read on page 560
- statistics add_metric on page 557

Affected by this attribute: statistics_enable_power_report
statistics reset

Removes the statistics information (QoR metrics) recorded or read in during the session.

Example

rc:/> statistics run_stage_ids

    Run & Stage ID summary
    ----------------------
    Run ID     Stage ID(s)  Run Description
    ----------------------
    sample2     elaborate  generic global_map incremental n/a

Info : Reset the statistics information preset in the database. [STAT-4]

rc:/> 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

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Affects these commands: statistics report on page 562

statistics run_stage_ids on page 566
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

Tracking and Saving QoR Metrics in Using Encounter RTL Compiler

Affected by these commands: statistics log on page 558
statistics read on page 560
statistics reset on page 565

Related attributes: statistics_run_id
statistics_run_description
Statistics write

The `statistics write` command can be used to write 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**

```plaintext
rc:/ > set_attribute statistics_db_file stats/test.db /
Setting attribute of root '/': 'statistics_db_file' = stats/test.db
rc:/ > statistics write
Info : Writing statistics database to file. [STAT-3]
    : Writing to db file 'stats/test.db'
```

**Related Information**

- **Tracking and Saving QoR Metrics** in *Using Encounter RTL Compiler*

  Related command: `statistics read` on page 560

  Related attribute: `statistics_db_file`
**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.
  
  ```
rc:/> 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.
  
  ```
rc:/> timestat
====================================================================
The RUNTIME after undefined is 0.45 secs
and the MEMORY_USAGE after undefined is 24.16 MB
====================================================================
```
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.
Examples

- The following command reads the Encounter Timing System commands to be executed from the ets_include file.

  ```bash
  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.

  ```bash
  validate_timing -rep_tim_str "report_timing"
  ```
Physical

- check_floorplan on page 573
- check_placement on page 577
- create_group on page 579
- create_placement_blockage on page 580
- create_placement_halo_blockage on page 581
- create_region on page 582
- create_row on page 584
- create_routing_blockage on page 585
- create_routing_halo_blockage on page 587
- create_track on page 588
- def_move on page 589
- duplicate_register on page 590
- generate_ple_model on page 592
- generate_reports on page 594
- modify_power_domain_attr on page 596
- move_blockage on page 598
- move_instance on page 599
- move_port on page 600
- move_region on page 601
- read_def on page 602
- read_encounter on page 606
Command Reference for Encounter RTL Compiler
Physical

- read_sdp_file on page 607
- read_spef on page 608
- report congestion on page 609
- report utilization on page 610
- resize_blockage on page 611
- resize_region on page 612
- restore_congestion_map on page 613
- save_congestion_map on page 614
- specify_cell_pad on page 615
- specify_floorplan on page 616
- summary_table on page 618
- update_congestion_map on page 620
- update_gcell_congestion on page 621
- update_gcell_pin_density on page 622
- update_gcell_utilization on page 623
- write_def on page 624
- write_sdp_file on page 625
- write_spef on page 626
check_floorplan

check_floorplan
   [-design design]
   [-macros | -ports | -utilization]
   [-detailed -max_per_violation integer]
   [-status] [-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
- **-status** Specifies to return the status (0 or 1) of the command.
  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.
-report_utilization

Reports details about the areas with unusual utilization.

Examples

The following example shows the status of the floorplan

```
rc:/> check_floorplan -status
0
```

The following command checks for unplaced macros.

```
rc:/> check_floorplan -macros
```

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.

```
rc:/> check_floorplan -utilization
```

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.

rc:/> 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
The following command shows the detailed violation list.

rc://> 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 result.
: 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
**check_placement**

check_placement  
    
    [-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 RC-P 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:

```
rc:/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:

```
rc:/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)
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
create_group

create_group group instance_list
    [-region region]
    [-design design]

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.
- `-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:

```shell
create_group mygroup -region myreg [find / -inst a_reg_3] [find / -inst a_reg_5]
```

**Related Information**

- Affects this command: `synthesize` on page 377
- Related command: `create_region` on page 582
create_placement_blockage

create_placement_blockage
   -boxes {llx lly urx ury}...
   [-component name]
   [-pushdown] [-partial] [-soft]
   [-density float] [-no_update] [-design design]

Creates a placement blockage.

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 to percentage of the blockage area to be used for standard cells during initial placement.
-design name  Specifies the name of the design for which to create the blockage.
-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.

Related Information

Affects this command:  
synthesize on page 377
Related commands:  
create_placement_halo_blockage on page 581
move_blockage on page 598
resize_blockage on page 611
create_placement_halo_blockage

create_placement_halo_blockage
- left integer - right integer
- top integer - bottom integer
[ -soft ] [ -no_update ] instance

Creates a placement halo blockage around the specified instance.

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.
- 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.
- 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 this command: synthesize on page 377
Related commands: create_placement_blockage on page 580
move_blockage on page 598
resize_blockage on page 611
create_region

create_region region
    [-fence | -guide ] [-no_update]
    (-boxes {llx lly urx ury}...| -polygon 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]...:
    Specifies a list of the coordinates of at least three points that define the region polygon. Use floats for the coordinates.

region:
    Specifies the name of the region to be created.

Example

The following command creates region my_reg of type fence for design DTMF:

create_region myreg -fence -box {200 200 1000 1000} [find / -design DTMF]

Related Information

Affects this command: \textit{synthesize} on page 377
Related commands:  
move_region on page 601  
resize_region on page 612

Related Attributes:  
Region Attributes
create_row

create_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:

```bash
create_row my_row -macro core_site -orientation N -llx 250 -lly 260 -width 900
    -height 5 -design [find / -design DTMF]
```

**Related Information**

- **Affects this command:** synthesize on page 377
- **Related Attributes:** Row Attributes
create_routing_blockage

create_routing_blockage
  -layer layer
  -boxes {llx lly urx ury}...
  [-slots] [-fills] [-pushdown]
  [-component name] [-no_update] [-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.

-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.
  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.
Related Information

Affects this command:  
synthesize on page 377

Related commands:  
create_routing_halo_blockage on page 587
move_blockage on page 598
resize_blockage on page 611
create_routing_halo_blockage

create_routing_halo_blockage
    -min_layer layer -max_layer layer
    -distance integer
    [-no_update] instance

Creates a routing halo blockage around the specified instance.

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.

-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.

Note: One DB unit is 1/1000 micron.

Related Information

Affects this command:          synthesize on page 377
Related command:               create_routing_blockage on page 585
                                move_blockage on page 598
                                resize_blockage on page 611
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.

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, while the Y coordinate applies to horizontal tracks.
   -step integer           Specifies the spacing between the tracks.
   -vertical               Specifies to create vertical tracks.
                           By default, a horizontal track is created.

Related Information

Related command:          synthesize on page 377
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 synthesize -to placed 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
synthesize -to_placed -incr
def_move -highlight

Related Information

Related command: synthesize on page 377
**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.
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
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 Encounter. 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 miscorrelation. The header might look like:

```
# DESIGN NAME: DTMF_CHIP
# TECHNOLOGY LEF: aII.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.

- **-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:

```plaintext
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:

```plaintext
set_attribute library library_list /
set_attribute lef_library lef_files /
read_hdl hdl_file
elaborate
decrypt ple_file
...
```

Related Information

Affects these commands: `synthesize -to_mapped`
`synthesize -toplaced`
generate_reports

```bash
generate_reports -outdir path -tag string
    [-encounter]
```

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**

- `-encounter` Specifies to use the QoS statistics from the latest Encounter run. By default, the statistics are used from RTL Compiler.

- `-outdir path` Specifies the path to the directory where the output data should be stored.

- `-tag string` Specifies the tag name for reports generated at this stage.

**Example**

Assume the following `generate_reports` are executed.

```bash
rc:/> generate_reports -outdir TEST_OUT -tag initial...
rc:/> generate_reports -outdir TEST_OUT -tag generic...
rc:/> 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
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 618
Affected by this attribute: qos_report_power
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.
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
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 this command: `synthesize` on page 377
- Related commands: `create_placement_blockage` on page 580, `create_placement_halo_blockage` on page 581, `resize_blockage` on page 611
move_instance

move_instance instance
   [-x integer | -dx integer]
   [-y integer | -dy integer]

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

- **-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:
  
```plaintext
  move_instance -dx 2 [find . -inst PLLCLK_INST]
```
move_port

move_port port
    [-x integer | -dx integer]
    [-y integer | -dy integer]

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

- 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.

Examples

- The following command moves port vdd2 up over 200 DB units in the vertical direction:

  move_port -dy 200 pllrst
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 this command:  synthesize on page 377
Related commands:  create_region on page 582
                   resize_region on page 612
read_def

loads the specified DEF file.

RTL Compiler 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.

RTL Compiler 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.

  def_file

  Specifies the DEF file.

- `-design design` Specifies the design to which to annotate the DEF information.

- `-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_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_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:

```
rc:/> 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/fool' 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
----
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
read_encounter

read_encounter config configuration_file

Reads an Encounter configuration file into RTL Compiler. An Encounter 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. Encounter 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

classification_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.

rc:/> read_encounter config fast.config
rc:/> synthesize -to_mapped
...

Related Information

Related command:  write_encounter on page 263
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 Design with RTL Compiler Physical

Related command: write_sdp_file on page 625
read_spef

read_spef spef_file
    [-max_fanout integer]
    [-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 RTL Compiler, the SPEF file is generated by Encounter.

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 is 1000.

spef_file        Specifies the SPEF file.

Related Information

Related command: write_spef on page 626
report congestion

Refer to report congestion in Chapter 9, “Analysis and Report.”
report utilization

Refer to report utilization in Chapter 9, “Analysis and Report.”
**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 this command: `synthesize` on page 377

Related commands: `create_placement_blockage` on page 580
`create_placement_halo_blockage` on page 581
`move_blockage` on page 598
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 this command: `synthesize` on page 377
- Related commands: `create_region` on page 582, `move_region` on page 601
**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 RC-P 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:  
- `save_congestion_map` on page 614
- `update_congestion_map` on page 620
save_congestion_map

save_congestion_map [-design design] file

Saves the congestion map to the specified file.

**Note:** This command applies only to the RC-P flow.

**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 613  
- `update_congestion_map` on page 620
**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 this command: `synthesize` on page 377
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.

  rc:/> specify_floorplan DTMF_CHIP -height 1400 -width 1200
  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.

- 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 this command: \texttt{synthesize} on page 377
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.

rc://> summary_table -outdir $env(TEST_OUT_DIR)

Working Directory = /...test
QoS Summary for cscan
==================================================================
<table>
<thead>
<tr>
<th>Metric</th>
<th>initial</th>
<th>generic</th>
<th>map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack (ps):</td>
<td>1958.6</td>
<td>1934.1</td>
<td>1190.1</td>
</tr>
<tr>
<td>R2R (ps):</td>
<td>no_value</td>
<td>no_value</td>
<td>no_value</td>
</tr>
<tr>
<td>I2R (ps):</td>
<td>1958.6</td>
<td>1934.1</td>
<td>1190.1</td>
</tr>
<tr>
<td>R2O (ps):</td>
<td>2152.6</td>
<td>2152.6</td>
<td>1987.2</td>
</tr>
<tr>
<td>I2O (ps):</td>
<td>no_value</td>
<td>no_value</td>
<td>no_value</td>
</tr>
<tr>
<td>CG (ps):</td>
<td>2100.0</td>
<td>2100.0</td>
<td>2100.0</td>
</tr>
<tr>
<td>TNS (ps):</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R2R (ps):</td>
<td>no_value</td>
<td>no_value</td>
<td>no_value</td>
</tr>
<tr>
<td>I2R (ps):</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R2O (ps):</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I2O (ps):</td>
<td>no_value</td>
<td>no_value</td>
<td>no_value</td>
</tr>
<tr>
<td>CG (ps):</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Failing Paths:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Area:</td>
<td>36</td>
<td>28</td>
<td>43</td>
</tr>
<tr>
<td>Instances:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Utilization (%):</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Tot. Net Length (um): no_value no_value no_value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. Net Length (um): no_value no_value no_value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Overflow H: 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Overflow V: 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route Overflow H (%): no_value no_value no_value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route Overflow V (%): no_value no_value no_value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
==================================================================
| CPU Runtime (m:s): | 00:04 | 00:00 | 00:01 |
| Real Runtime (m:s): | 00:13 | 01:35 | 01:30 |
Related Information

Affected by this command: generate_reports on page 594

Affected by this attribute: qos_report_power
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 RC-P 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.
update_gcell_congestion

update_gcell_congestion [-design design]

Updates the congestion values of the gcells for the specified design. You can run this command after any command that updates the physical data.

**Note:** This command applies only to the RC-P flow.

**Options and Arguments**

- **-design design** Specifies the design for which the gcell congestion values must be updated.
update_gcell_pin_density

update_gcell_pin_density [-design design]

Updates the pin density values of the gcells for the specified design. You can run this command after any command that updates the physical data.

Note: This command applies only to the RC-P flow.

Options and Arguments

-design design Specifies the design for which the gcell pin density values must be updated.
**update_gcell_utilization**

update_gcell_utilization [-design design]

Updates the utilization values of the gcells for the specified design. You can run this command after any command that updates the physical data.

**Note:** This command applies only to the RC-P flow.

**Options and Arguments**

- `-design design` Specifies the design for which the gcell utilization values must be updated.
write_def

```
write_def [-ignore_groups] [-ignore_placed_instances] 
    [-hierarchical instance] [-scan_chains] 
    [-version string] [design] [> file]
```

Writes a floorplan, in DEF format, for the specified design. The DEF does not contain the 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.

**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.
- **-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
write_sdp_file

write_sdp_file
    [-file file]
    [-top_group sdp_group]

Saves the current SDP relative placement information in the specified file.

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 Design with RTL Compiler Physical

Related command: read_sdp_file on page 607
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 608
Design for Test

- add_opcg_hold_mux on page 632
- analyze_scan_compressibility on page 633
- analyze_testability on page 642
- check_atpg_rules on page 645
- check_dft_pad_configuration on page 647
- check_dft_rules on page 648
- check_mbist_rules on page 654
- compress_block_level_chains on page 657
- compress_scan_chains on page 660
- concat_scan_chains on page 675
- configure_pad_dft on page 677
- connect_compression_clocks on page 678
- connect_opcg_segments on page 679
- connect_scan_chains on page 681
- define_dft on page 686
- define_dft_abstract_segment on page 689
- define_dft_boundary_scan_segment on page 695
- define_dft_dft_configuration_mode on page 699
- define_dft_domain_macro_parameters on page 702
- define_dft_fixed_segment on page 704
- define_dft_floating_segment on page 706
- `define_dft jtag_instruction` on page 708
- `define_dft jtag_instruction_register` on page 712
- `define_dft jtag_macro` on page 714
- `define_dft mbist_clock` on page 719
- `define_dft mbist_direct_access` on page 722
- `define_dft opcg_domain` on page 725
- `define_dft opcg_mode` on page 728
- `define_dft opcg_trigger` on page 730
- `define_dft osc_source` on page 732
- `define_dft pmbist_direct_access` on page 734
- `define_dft preserved_segment` on page 736
- `define_dft scan_chain` on page 739
- `define_dft scan_clock_a` on page 745
- `define_dft scan_clock_b` on page 748
- `define_dft shift_enable` on page 751
- `define_dft shift_register_segment` on page 754
- `define_dft tap_port` on page 756
- `define_dft test_bus_port` on page 758
- `define_dft test_clock` on page 762
- `define_dft test_mode` on page 766
- `dft_trace_back` on page 770
- `fix_dft_violations` on page 772
- `fix_scan_path_inversions` on page 776
- `identify_domain_crossing_pins_for_cgic_and_scan_abstracts` on page 777
- `identify_multibit_cell_abstract_scan_segments` on page 778
- `identify_shift_register_scan_segments` on page 780
- `identify_test_mode_registers` on page 782
- `insert_dft` on page 785
- `insert_dft boundary_scan` on page 788
- `insert_dft compression_logic` on page 793
- `insert_dft dfa_test_points` on page 804
- `insert_dft jtag_macro` on page 808
- `insert_dft lookup_element` on page 812
- `insert_dft logic_bist` on page 813
- `insert_dft mbist` on page 819
- `insert_dft opcq` on page 825
- `insert_dft pmbist` on page 827
- `insert_dft ptam` on page 833
- `insert_dft rra_test_points` on page 836
- `insert_dft scan_power_gating` on page 843
- `insert_dft shadow_logic` on page 846
- `insert_dft shift_register_test_points` on page 851
- `insert_dft test_point` on page 852
- `insert_dft user_test_point` on page 858
- `insert_dft wir_signal_bits` on page 860
- `insert_dft wrapper_cell` on page 861
- `insert_dft wrapper_instruction_register` on page 867
- `insert_dft wrapper_mode_decode_block` on page 869
- `insert_test_compression` on page 871
- `map_mbist_cgc_to_cgic` on page 874
- `read_dft_abstract_model` on page 875
- `read_io_speclist` on page 878
- `read_memory_view` on page 879
- `read_pmbist_interface_files` on page 881
■ replace_opcg_scan on page 882
■ replace_scan on page 884
■ report_dft_chains on page 885
■ report_dft_clock_domain_info on page 886
■ report_dft_coreWrapper on page 887
■ report_dft_registers on page 888
■ report_dft_setup on page 889
■ report_dft_violations on page 890
■ report_opcg_equivalents on page 891
■ report_scan_compressibility on page 892
■ report_test_power on page 893
■ reset_opcg_equivalent on page 894
■ reset_scan_equivalent on page 895
■ set compatible_test_clocks on page 896
■ set_opcg_equivalent on page 898
■ set_scan_equivalent on page 900
■ update_scan_chains on page 902
■ write_atpg on page 904
■ write_bsd1 on page 907
■ write_compression_macro on page 910
■ write_dft_abstract_model on page 918
■ write_dft_rtl_model on page 921
■ write_et_atpg on page 922
■ write_et_bsv on page 930
■ write_et_dfa on page 934
■ write_et_mbist on page 938
■ write_et_mbist on page 942
write_et_no_tp_file on page 947
write_et_rrfa on page 948
write_io_speclist on page 952
write_logic_bist_macro on page 954
write_mbist_testbench on page 958
write_pmbist_interface_files on page 962
write_pmbist_testbench on page 964
write_scandef on page 968
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-equivalency mappings with the set_opcg_equivalent command. If the OPCG-equivalency mappings are not available, or if the scan cell type of the instance does not have an entry in the OPCG-equivalency table, the command adds a hold mux before the scan cell to convert it to an OPCG-equivalent cell and makes all necessary connections.

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 894
set_opcg_equivalent on page 898
analyze_scan_compressibility

analyze_scan_compressibility
   -library string [design] [-chains integer]
   [-minimum_scanned_flop_percentage integer]
   [-compressor {xor | misr | mimic_bidi_misr}]
   [-decompressor {broadcast | xor}]
   [-mask {widel | wide0 | wide2}] [-serial_misr]
   [-fault_sample_size fault_sample_size]
   [-ratio_list list_of_compression_ratios]
   [-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.

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 -chains option.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

-atpg_options string

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 Encounter Test 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 Encounter Test documentation).

-build_model_options string

Specifies extra options to apply when building the Encounter Test 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 Encounter Test documentation).

-build_testmode_options string

Specifies extra options to apply when building the test mode for Encounter Test.

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 *Command Line Reference* (of the Encounter Test documentation).

-chains integer

Specifies the number of scan chains to be analyzed. This option is only required if no actual or user-defined scan chains exist.

The specified number must be equal to or larger than one.

-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
- **mimic_bidi_misr** analyzes an on-product MISR compressor 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:** broadcast

**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.

**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_size integer_sample_size

Specifies the number of faults to simulate to predict test coverage. This number affects the number of partitions or slices to be run depending on the total number of faults in the design. Specifying a higher number obtains more accurate results while specifying a lower number reduces runtime.

**Defaults:** 20000. For full ATPG, -1.

-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:

   ```
   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 Encounter Test will be run.

-mask {wide1 | wide0 | wide2}

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, no masking logic is inserted.
RTL Compiler 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 list_of_compression_levels

Specifies the list of compression ratios to be analyzed.

**Default:** "20 50 100".

### -serial_misr_read

Performs analysis for serial MISR compression.

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 for Encounter Test.

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 *Command Line Reference* (of the Encounter Test 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.

rc:/> 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
 Decompresor - broadcast
 Compressor   - xor
 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

Encounter Test 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
Encounter Test 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

Encounter Test 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

Encounter Test scripts for ATPG have been written for compression mode with fullscan top-off to directory '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 a broadcast decompressor, and a ratio list of "10 20 30".

rc:> 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'

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
Encounter Test 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
Encounter Test 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
Encounter Test 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
Encounter Test scripts for ATPG have been written for compression mode with fullscan top-off to directory './newdirnr2/et_scripts_ratio_30'.

Running Encounter Test 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
----------------------------------------
Achieved compression table with fullscan top-off vectors

<table>
<thead>
<tr>
<th>IC</th>
<th>TATR</th>
<th>TDVR</th>
<th>ATCov.</th>
<th>CL</th>
<th>PAT-comp</th>
<th>PAT-fs</th>
<th>Cycles</th>
<th>Runtime</th>
<th>Gatecount</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>1.0</td>
<td>1.0</td>
<td>99.9%</td>
<td>1348</td>
<td>-</td>
<td>305</td>
<td>414762</td>
<td>00:00:03</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>5.6</td>
<td>5.6</td>
<td>99.72%</td>
<td>135</td>
<td>301</td>
<td>22</td>
<td>73532</td>
<td>00:00:04</td>
<td>55</td>
<td>680.656999999992</td>
</tr>
<tr>
<td>20</td>
<td>5.5</td>
<td>5.5</td>
<td>99.45%</td>
<td>68</td>
<td>324</td>
<td>37</td>
<td>75711</td>
<td>00:00:05</td>
<td>112</td>
<td>1357.9199999999983</td>
</tr>
<tr>
<td>30</td>
<td>4.4</td>
<td>4.4</td>
<td>97.83%</td>
<td>45</td>
<td>299</td>
<td>57</td>
<td>95084</td>
<td>00:00:13</td>
<td>155</td>
<td>1941.8260000000001</td>
</tr>
</tbody>
</table>

Achieved compression table without fullscan top-off vectors

<table>
<thead>
<tr>
<th>IC</th>
<th>TATR</th>
<th>TDVR</th>
<th>ATCov.</th>
<th>CL</th>
<th>PAT-comp</th>
<th>PAT-fs</th>
<th>Cycles</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>1.0</td>
<td>1.0</td>
<td>99.9%</td>
<td>1348</td>
<td>-</td>
<td>305</td>
<td>414762</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9.8</td>
<td>9.8</td>
<td>99.72%</td>
<td>135</td>
<td>301</td>
<td>-</td>
<td>42212</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>17.6</td>
<td>17.6</td>
<td>99.45%</td>
<td>68</td>
<td>324</td>
<td>-</td>
<td>23586</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>27.8</td>
<td>27.8</td>
<td>97.83%</td>
<td>45</td>
<td>299</td>
<td>-</td>
<td>14946</td>
<td></td>
</tr>
</tbody>
</table>

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.

```
rc:> 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 *Design for Test in Encounter RTL Compiler*.

Affected by these commands:  
- `connect_scan_chains` on page 681  
- `define_dft_scan_chain` on page 739

Related command:  
- `report_scan_compressibility` on page 529
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]
[-etlog file] -directory string [design]

Invokes Encounter Test to perform Automatic Test Pattern Generator (ATPG) based testability analysis in either assume or fullscan mode.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in *Design for Test in Encounter RTL Compiler*.

**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 Encounter Test 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 Encounter Test documentation).

- **-build_model_options string**
  Specifies a string containing the extra options to build a model.
  
  **Note:** For more information on these options, refer to the build_model command in the *Command Line Reference* (of the Encounter Test 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 Encounter Test 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 Encounter Test.

-et_log file

Specifies the name of the Encounter Test log file. This file will
be generated in the specified directory.

Default: eta_from_rc.log

-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.
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"
  ```

Related Information

Using Encounter Test Software to Analyze Testability in Design for Test in Encounter RTL Compiler.

Affected by these constraints:  

- `define_dft test_mode` on page 766
- `define_dft test_clock` on page 762
check_atpg_rules

check_atpg_rules [-library string] [-compression] [-directory string] [design]

Generates a template script to run Encounter Test 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:

- et.exclude—A file listing objects to be excluded from the ATPG analysis

- et.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
runet.tsv—A template script file for Encounter Test 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 Encounter Test.
  
  **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.

Example

```bash
rc:/> check_atpg_rules
```

Encounter Test scripts to check whether a design is ATPG ready have been written to directory './check_atpg_rules'.

Invoke the scripts as 'et -e ./check_atpg_rules/runet.tsv'
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.

Related Information

Checking the Pad Configuration in Design for Test in Encounter RTL Compiler.
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.

Table 11-1 lists the DFT rule violations that this command checks.

Table 11-1  Checking For and Auto-Fixing of DFT Rule Violations

<table>
<thead>
<tr>
<th>DFT Rule Violation</th>
<th>Checked?</th>
<th>Auto-Fixed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncontrollable asynchronous set or reset signals</td>
<td>Checked</td>
<td>Yes</td>
</tr>
<tr>
<td>Gated clocks and derived clocks</td>
<td>Checked</td>
<td>Yes</td>
</tr>
<tr>
<td>Flip-flop's clock port connected to tied lines</td>
<td>Checked</td>
<td>Yes</td>
</tr>
<tr>
<td>Conflicting clock and asynchronous set or reset signals</td>
<td>Checked</td>
<td>No</td>
</tr>
<tr>
<td>Tristate contention for internal and external nets</td>
<td>Checked(*)</td>
<td>Yes</td>
</tr>
<tr>
<td>Same asynchronous set or reset data race conditions</td>
<td>Checked(*)</td>
<td>Yes</td>
</tr>
<tr>
<td>Clock and data race conditions</td>
<td>Checked(*)</td>
<td>No</td>
</tr>
<tr>
<td>Floating nets violations</td>
<td>Checked(*)</td>
<td>No</td>
</tr>
<tr>
<td>X-source violations</td>
<td>Checked(*)</td>
<td>Yes</td>
</tr>
<tr>
<td>Floating conditions</td>
<td>Not checked</td>
<td>No</td>
</tr>
</tbody>
</table>

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.
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.

- **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.

  rc:/> define_dft shift_enable scan_en -active high
  rc:/> 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
  Percentage of total registers that are scannable: 20%
  rc:/> 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.

rc:/> 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'.

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.
---------------------
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- 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 689  
- `define_dft dft_configuration_mode` on page 699  
- `define_dft shift_enable` on page 751  
- `define_dft test_clock` on page 762  
- `define_dft test_mode` on page 766

Affects these commands:  
- `connect_scan_chains` on page 681  
- `fix_dft_violations` on page 772  
- `report dft_registers` on page 888  
- `synthesize` on page 377

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`  
- `dftViolation`  
- `type`  
- `Violations Attributes`
check_mbist_rules

check_mbist_rules [-design design]
                  [-dft_configuration_mode dft_config_mode_name]
                  [-direct_access_only] [-interface_file_dirs string]

Checks for MBIST rule violations on the specified design.

Table 11-2 lists the MBIST rules checked by the command. The command does not auto-fix detected MBIST rules violations.

Table 11-2  Checked MBIST Rules

<table>
<thead>
<tr>
<th>MBIST Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ The Test_Control test signal used during insert dft_mbist is properly</td>
</tr>
<tr>
<td>controlled at the pin of all MBIST engines when the dft_configuration_mode</td>
</tr>
<tr>
<td>is active.</td>
</tr>
<tr>
<td>■ All MBIST engine clock pins are properly controlled from design ports or</td>
</tr>
<tr>
<td>internal pins for MBIST clocks defined with the define_dft mbist_clock</td>
</tr>
<tr>
<td>-internal_clock_source command</td>
</tr>
<tr>
<td>■ All MBIST interface files are consistent with the JTAG instructions and</td>
</tr>
<tr>
<td>MBIST logic inserted in the design. Portions of the design which are</td>
</tr>
<tr>
<td>blackboxes containing MBIST logic are limited to interface checks.</td>
</tr>
<tr>
<td>■ The need for a user supplied mode initialization sequence by downstream</td>
</tr>
<tr>
<td>tools in case MBIST direct access is implemented using internal design pins</td>
</tr>
<tr>
<td>as control signals.</td>
</tr>
</tbody>
</table>

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

- **-design design** Specifies the name of the top-level design to be checked.
- **-dft_configuration_mode dft_configuration_mode_name**
  Specifies the configuration mode in which netlist tracing must be done.
  
  *Default*: mbist
-direct_access_only Specifies that the MBIST logic was inserted using the MBIST direct access control method, and no JTAG macro for MBIST operations.

Default: the design of the current directory in which the command is executed.

-interface_file_dirs dir1 dir2 ...

Checks the content of MBIST interface files against the MBIST logic inserted into the design.

Default: value of the directory keyword for each design when the insert_dft mbist command was executed.

Example

The following example shows how to define and use an MBIST configuration mode.

# Define the test signal states needed to establish the fullscan (ATPG) mode \
# of operation
define_dft test_mode -name TE -active high TESTENABLE
define_dft test_mode -name TM0 -active high PORT00
define_dft test_mode -name TM1 -active low PORT01
define_dft test_mode -name TM2 -active low PORT02
define_dft test_mode -name TM3 -active high PORT03
define_dft test_mode -name TM4 -active low PORT04

# Define the test signal states needed to establish the MBIST mode of operation
define_dft dft_configuration_mode -name My_MBIST_Mode \
    -mode_enable_high TE TM0 TM1 TM2 -mode_enable_low TM3 TM4

# Use the MBIST configuration mode to insert the MBIST logic and verify 
# the MBIST rules
insert_dft mbist ... -directory ./My_MBIST_Dir \
    -dft_configuration_mode My_MBIST_Mode -test_control TM3
check_mbist_rules -dft_configuration_mode My_MBIST_Mode \
    -interface_file_dirs ./My_MBIST_Dir

Note: The logic states of the test signals specified in the MBIST configuration mode are decoded in the user-created logic and together with the test control signal of the insert dft_mbist command establish the MBIST mode of operation.
Related Information

Design Flows in “Inserting Memory Built-In-Self-Test Logic” in Design for Test in Encounter RTL Compiler

Affected by these constraints:

- `define_dft dft_configuration_mode` on page 699
- `define_dft mbist_clock` on page 719
- `define_dft mbist_direct_access` on page 722
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}] ]
  [-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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

**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:
  - xor specifies to build an XOR-based compressor
misr specifies to build a MISR-based 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.

-power_aware

Ensures that the mux logic added during compression obeys the boundaries of the power domains defined in the CPF file.
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

Hierarchical Compression Flow in Design for Test in Encounter RTL Compiler

Affected by these constraints:
- define_dft_jtag_instruction on page 708
- define_dft_shift_enable on page 751
- define_dft_test_mode on page 766

Affected by these commands:
- connect_scan_chains on page 681

Affects these commands:
- report_dft_chains on page 885
- write_dft_abstract_model on page 918

Sets these attributes:
- compressed
- dft_compression_signal
- dft_mask_clock
- dft_misr_clock
- type
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
            [-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 {wide0 | wide1 | wide2}]
        [-misr_charging_ratio integer]
        [-mask_clock {port|pin} [-allow_shared_clock]]
        [-mask_load test_signal]
        [-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}]
        [-misr_charging_ratio integer]
        [-mask_clock {port|pin} [-allow_shared_clock]]
        [-mask_load test_signal]
        [-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_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.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in *Design for Test in Encounter RTL Compiler*.

### 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 Encounter Test.

**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.
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, prefixcompression_enable, where prefix is the value of the dft_prefix root attribute.
-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.
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.
- **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.

  **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.

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.
-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.
-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).
-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.
-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.
-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 Encounter Test (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.
```
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.

rc:/> 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)
...```
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.

```
rc:/> 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.

```
rc:/> 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.

```
rc:/> 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

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Inserting Test Compression Logic
- Low Pin Count Compression Using Encoded Compression Signals
- Reducing Pin Count for Compression
Using Asymmetrical Scan Compression

Affected by these constraints:

- define_dft_jtag_instruction on page 708
- define_dft_shift_enable on page 751
- define_dft_test_mode on page 766

Affected by these commands:

- connect_scan_chains on page 681

Affects this command:

- report_dft_chains on page 885

Sets these attributes:

- compressed
- dft_compression_signal
- dft_mask_clock
- dft_misr_clock
- type
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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **Concatenating Scan Chains**
- **Controlling Scan Configuration**

Affected by these constraints:
- `define_dft dft_configuration_mode` on page 699
- `define_dft shift_enable` on page 751
- `define_dft test_mode` on page 766
- `set_compatible_test_clocks` on page 896

Affected by this command:
- `check_dft_rules` on page 648

Affects these commands:
- `report dft_chains` on page 885
- `write_dft_abstract_model` on page 918
- `write_scandef` on page 968

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`
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.

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 RC-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

See the following sections in Design for Test in Encounter RTL Compiler:

- Configuring Bidirectional and Tristate Pads in Test Mode
- Using a Functional Data Pin to Drive a Shift-Enable Test Signal

Affected by these constraints: define_dft shift_enable on page 751
   define_dft test_mode on page 766

Affects these commands: check_dft_rules on page 648
   connect_scan_chains on page 681
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.

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.

Related Information

Hierarchical Compression Flow in Design for Test in Encounter RTL Compiler

Affected by these commands: compress_block_level_chains on page 657
compress_scan_chains on page 660
connect_opcg_segments

connect_opcg_segments
  [-chains actual_scan_chains]
  [-use_ports_for_side_scan_connections port_list]
  [-preview] [design]

Connects the scan segments associated with the OPCG logic into the actual scan chains. You must run this command after you have connected the scan chains.

Options and Arguments

**design**
Specifications 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**

Specifications 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.

**-use_ports_for_side_scan_connections port_list**

Specifications the ports to be used as scan data input pins when making the connections for the side scan chains.

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

Connecting the OPCG Segments in Design for Test in Encounter RTL Compiler

Affected by these commands

- connect_scan_chains on page 681
- insert_dft_opcg on page 825
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]
  [-power_domain power_domain_list] [-update_placement]

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.
-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.

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.

- **-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.

- **-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 `synthesize -to_placed`. The option will be ignored if the design is not placed.

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Controlling Scan Configuration
- Connecting the Scan Chains
- Library-Domain Aware Scan Chain Configuration
- Power-Domain Aware Scan Chain Configuration
- Physical Scan Chain Synthesis
- Defining Scan Configuration Modes

Related commands: `write_compression_macro` on page 910

Affected by these constraints: `define_dft abstract_segment` on page 689
`define_dft dft_configuration_mode` on page 699
`define_dft fixed_segment` on page 704
`define_dft floating_segment` on page 706
`define_dft preserved_segment` on page 736
`define_dft scan_chain` on page 739
`define_dft scan_clock_a` on page 745
`define_dft scan_clock_b` on page 748
`define_dft shift_enable` on page 751
`define_dft shift_register_segment` on page 754
`set_compatible_test_clocks` on page 896
Affected by these commands:

- **check_dft_rules** on page 648
- **fix_dft_violations** on page 772
- **synthesize** on page 377

Affects this command:

- **concat_scan_chains** on page 675
- **report_dft_chains** on page 885

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**
**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 | mbist_direct_access  
| 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.
- **mbist_direct_access**: Defines MBIST direct access interface pins or ports.
- **opcg_domain**: Defines an OPCG (clock) domain.
- **opcg_mode**: Defines an OPCG mode for Encounter Test ATPG.
Related Information

Related commands:

- `define_dft abstract_segment` on page 689
- `define_dft boundary_scan_segment` on page 695
- `define_dft dft_configuration_mode` on page 699
- `define_dft domain_macro_parameters` on page 702
- `define_dft fixed_segment` on page 704
- `define_dft floating_segment` on page 706
- `define_dft jtag_instruction` on page 708
- `define_dft jtag_instruction_register` on page 712
- `define_dft jtag_macro` on page 714
define_dft mbist_clock on page 719
define_dft mbist_direct_access on page 722
define_dft opcg_domain on page 725
define_dft opcg_mode on page 728
define_dft opcg_trigger on page 730
define_dft osc_source on page 732
define_dft pmbist_direct_access on page 734
define_dft preserved_segment on page 736
define_dft scan_chain on page 739
define_dft scan_clock_a on page 745
define_dft scan_clock_b on page 748
define_dft shift_enable on page 751
define_dft shift_register_segment on page 754
define_dft tap_port on page 756
define_dft test_bus_port on page 758
define_dft test_clock on page 762
define_dft test_mode on page 766
**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}]  
[-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.

-`-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.

**-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.

**-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.

- **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.

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.
  
  ```
  rc:/> 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.
  
  ```
  rc:/> 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
  ```

### Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Defining Abstract Segments
- Using Abstract Segments
- Creating Head, Body, and Tail Segments

Affects this constraint: define_dft dft_configuration_mode on page 699

Affects these commands: check_dft_rules on page 648
report dft_chains on page 885

Sets these attributes: Scan Segment Attributes
**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.

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_ACPCLK` 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`  
  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
    Specifies the name of the MODE_A pin on the boundary scan segment.
Example

- The following example defines an boundary scan segment using a set of differential port pairs.

```
rc:/ 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.
  - 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.

```shell
rc:/> 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**

**Defining Boundary Scan Segments** in *Design for Test in Encounter RTL Compiler*

Affects these commands:
- `insert_dft_boundary_scan` on page 788
- `write_bsdl` on page 907

Sets these attributes:
- `bcell_segment`
- `differential`
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.

Note: To use this command option you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

-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.
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

See the following sections in *Design for Test in Encounter RTL Compiler*

- Defining Scan Chain Configuration Modes
- JTAG-Controlled Scan Modes
- Defining the Wrapper Configuration Modes
- Inserting Core Wrapper Logic

Affects these commands:

- `report dft_chains` on page 448
- `check_dft_rules` on page 648
- `concat_scan_chains` on page 675
- `connect_scan_chains` on page 681
- `define_dft abstract_segment` on page 689
- `write_atpg` on page 904
write_dft_abstract_model on page 918
write_et_atpg on page 922
write_et_bsv on page 930
write_et_mbist on page 942
write_et_rrfa on page 948
write_scandef on page 968

Affects these attributes: DFT Configuration Mode Attributes
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_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

-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**

**Defining OPCG Domain Macro Parameters** in *Design for Test in Encounter RTL Compiler*

Affects this constraint: define_dft opcg_domain on page 725

Affects this command: insert_dft opcg on page 825

Sets these attributes: Domain Macro Parameters Attributes
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 739.

  ```
  define_dft fixed_segment -name segBody *seq/out_reg_1 *seq/out_reg_3
  ```
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

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Creating Head, Body, and Tail Segments
- Segmentation Rules in “Exporting the Design”

Affects this constraint: define_dft scan_chain on page 739
Affects these commands: connect_scan_chains on page 681
                      report dft_chains on page 885
Sets these attributes: Scan Segment Attributes
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 739.

rc:/designs/test> define_dft floating_segment -name segHead \
    *seq/out_reg_4 *seq/out_reg_5
rc:/designs/test> define_dft floating_segment -name segTail *seq/out_reg_0
Related Information

Creating Head, Body, and Tail Segments in *Design for Test in Encounter RTL Compiler*

Affects this constraint:  define _dft scan _chain on page 739
Affects these commands:  connect _scan _chains on page 681
                        report _dft _chains on page 885
Sets these attributes:   Scan Segment Attributes
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_reset_inverted]
   [-register_capturedr {pin|port}]
   [-register_clockdr {pin|port}]
   [-register_updatedr {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.

-`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_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_DECODER_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}
  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_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_registers_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.

-tap_decode {pin|port}

Specifies the name of the instruction-specific decode pin that must be created on the JTAG_MACRO subdesign.

-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

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Defining the Instructions
- Inserting a JTAG Macro
- Inserting Memory Built-In-Self-Test Logic
- JTAG-Controlled Scan Modes

Affects these commands:

- `insert_dft boundary_scan` on page 788
- `insert_dft jtag_macro` on page 808
- `insert_dft mbist` on page 819

Related command:

- `define_dft jtag_instruction_register` on page 712

Sets these attributes:

- JTAG Instruction Attributes
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 INSTRREGISTER 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 INSTRREGISTER -length 3
Related Information

See the following sections in Design for Test in Encounter RTL Compiler:

- Defining the Instruction Register
- Inserting a JTAG Macro
- Inserting Memory Built-In-Self-Test Logic
- JTAG-Controlled Scan Modes

Related commands: define_dft jtag_instruction on page 708
Affects these commands: insert_dft boundary_scan on page 788
insert_dft jtag_macro on page 808
insert_dft mbist on page 819
Related attributes: JTAG Instruction Register Attributes
define_dft jtag_macro

Identifies a pre-instantiated JTAG Macro.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

**Options and Arguments**

- **boundary_tdo** {pin|port|subport}
  
  Specifies the boundary register TDO output pin on the JTAG macro.

- **bsr_clockdr** {pin|port|subport}
  
  Specifies the clock data register (CLOCKDR) output pin for the boundary-scan register.

```plaintext
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}]
[-shiftdr {pin|port|subport}]
[-clockdr {pin|port|subport}]
[-updatedr {pin|port|subport}]
[-capturedr {pin|port|subport}]
[-exitdr_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_trcell_enable {pin|port|subport}]
[-por {pin|port|subport}] [-name string]
```
-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.

-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}  
   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 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.
Examples

- The following example defines a third party TAP controller with a specified instance location.

```bash
rc:/> 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 \ 
    -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.

```bash
rc:/> 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

See the following sections in *Design for Test in Encounter RTL Compiler*

- **Defining a Pre-Existing JTAG Macro**
- **JTAG-Controlled Scan Modes**

Sets these attributes: **JTAG Macro Attributes**

Related command: **insert_dft_jtag_macro** on page 808
define_dft mbist_clock

define_dft mbist_clock -name mbist_clock
    [-design design] [-internal_clock_source]
    -period integer
    [-hookup_pin pin ] [-hookup_period integer]
    [-hookup_polarity {non_inverted | inverted}]
    [-is_jtag_tck] [is_srcclk] 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. For more information, refer to Custom Boundary-Scan Cells in the Design for Test in Encounter RTL Compiler

The command returns the 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.

- **-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.
  
  **Note:** When you specify this option, the RC-DFT engine does not validate the controllability of any logic between the top-level MBIST (PMBIST) clock port and its designated hookup pin under test-mode setup.
-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.
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.

```bash
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

See the following sections in *Design for Test in Encounter RTL Compiler*:

- MBIST Clocking in “Inserting Memory Built-In-Self-Test Logic”
- Design Flows in “Inserting Memory Built-In-Self-Test Logic”

Affects these commands:

- `check_mbist_rules` on page 654
- `insert_dft mbist` on page 819
- `insert_dft pmbist` on page 827

Sets these attributes:

- MBIST Clock Attributes
define_dft mbist_direct_access

define_dft mbist_direct_access -function string
   -active {low | high}
   [-mtclk] {port | pin}

Defines MBIST direct access interface pins or ports which can be used as an alternative
access mechanism for MBIST. This method can be used as a supplement to or replacement
for the JTAG interface for controlling MBIST.

All signals are level-sensitive. The patterns that you generate by running the
create_embedded_test command in Encounter Test, properly stimulate and monitor the
defined signals if they are directly accessible from design ports.

Execution scheduling options are limited to all MBIST controllers running in parallel or serially
and the devices assigned to each engine running in parallel or serially.

Options and Arguments

   -active {low|high}  Specifies the active value of the signal.
   -function string   Defines the functionality of the direct access pin. The option can
                      have one of the following values:

                      ■ burnin_run

                      This signal must be held inactive for a minimum of three
                      MBIST clock periods at the start of the test sequence.
                      Once activated, the burnin_run signal causes the MBIST
                      operations to execute continuously within a loop until the
                      signal is deactivated. The results of the operations are
                      made visible on the monitor during the course of the
                      execution.

                      ■ device_schedule_serial

                      Causes the execution of devices within each MBIST
                      controller to occur serially when the signal is active. This
                      signal must remain stable throughout the test sequence.
                      The execution occurs in parallel when the signal is inactive
                      or undefined.
- **engine_schedule_serial**
  Causes the execution of MBIST controllers in the design to occur serially. This signal must remain stable throughout the test sequence.
  The execution occurs in parallel when the signal is inactive or undefined.

- **monitor**
  This signal is the logical AND of all MBIST controller done indications and inverted summary failure indications during a `poweron_run` operation. During a `burnin_run` operation, the signal is the logical AND of all MBIST inverted summary failure indications.

  **Note:** This function can also be used by a JTAG controller to monitor MBIST operations. You can specify it as the only direct access function in cases where the JTAG is the only access mechanism used for MBIST. In these circumstances, the monitor connection must not be shared and must not require any gating condition to enable observation.

- **poweron_run**
  This signal must be held inactive for a minimum of three MBIST clock periods at the start of the test sequence. When activated, a single execution of the MBIST operations occurs and the results of the operations are made visible on the monitor at completion.

- **-mtclk**
  Specifies whether the alternate `mtclk` clock input is used during direct access as applied to `burnin_run` or `poweron_run` functions.

  `{pin | port}`
  Specifies the source pin or port of the signal.
  A pin cannot have the `inout` direction if is used to monitor.
  A port cannot have the `inout` direction if is used to monitor and you use the MBIST block insertion flow.
  Any pin or port can be shared for functional uses provided the MBIST DFT configuration mode enables memory BIST usage during MBIST operations.
A pin specification for the monitor function implies that an internal monitor point is requested. The tool will not perform a forward trace to look for a port.

Example

The following example defines a power-on activated memory BIST with monitor and default execution schedule of parallel engine, parallel device.

```
define_dft mbist_direct_access -function poweron_run -active high \ports_in/poweron_mbist
define_dft mbist_direct_access -function monitor -active low \ports_out/DFT_scan_out[7]
```

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **RTL Compiler Prerequisites** in “Inserting Memory Built-In-Self-Test Logic”
- **Design Flows** in “Inserting Memory Built-In-Self-Test Logic”

Affects these commands: check_mbist_rules on page 654

```
insert_dft_mbist
```

Sets these attributes: Direct Access Function Attributes
define_dft opcg_domain

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

-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.

The domain macro parameter set must have been previously defined by the define_dft domain_macro_parameters command.

-location {subport | port | pin} Specifies where to insert the domain macro. Specify an existing top-level port, subport, or hierarchical pin name.

-min_domain_period float Specifies the minimum period (in picoseconds) at which the OPCG domain is assumed to operate.
**Command Reference for Encounter RTL Compiler**

**Design for Test**

---

**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.

- **-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.

**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`.

```plaintext
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`.

```plaintext
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`.

```plaintext
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`.

```plaintext
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

**Defining OPCG Clock Domains in** *Design for Test in Encounter RTL Compiler*

Affected by these constraints:  
- `define_dft domain_macro_parameters` on page 702  
- `define_dft opcg_trigger` on page 730  
- `define_dft osc_source` on page 732  

Affects this command:  
- `insert_dft opcg` on page 825  

Sets these attributes:  
- OPCG Domain Attributes
define_dft opcg_mode

define_dft opcg_mode [-name name]
  -mode_init file
  [-jtag_controlled]
  -osc_source_parameters string
  [-osc_source_parameters string]...
  [-design design]

Defines an OPCG mode for Encounter Test 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.
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 opcgModel.mode_init
```

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Defining the OPCG Mode
- Writing the Scripts and Setup Files to Perform ATPG
- Generate Files for ATPG and Simulation

Affects this command:  
```
write_et_atpg
```
on page 922

Related constraint:  
```
define_dft osc_source
```
on page 732

Sets these attributes:  
```
OPCG Mode Attributes
```

```
Osc Source Reference Attributes
```
### define_dft opcg_trigger

```c
define_dft opcg_trigger [-name name] 
[-active {low|high}]
-osc_source osc_source 
[{pin|port}] [-create_port]
[-inside hierarchical_instance]
[-scan_clock scan_clock] 
[-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.

- **-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.

  **Default:** inserted at the top-level of the design.

- **-name name**
  
  Specifies the `opcg_trigger` object name of the OPCG trigger signal.

  **Default:** OPCG_TRIGGER_n
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

**Defining OPCG Triggers** in *Design for Test in Encounter RTL Compiler*

- Affects this command: `insert_dft opcg` on page 825
- Related constraints: `define_dft osc_source` on page 732, `define_dft opcg_domain` on page 725, `define_dft opcg_mode` on page 728
- Sets these attributes: `OPCG Trigger Attributes`
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.
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

Defining the Oscillator Sources in *Design for Test in Encounter RTL Compiler*

Affects these commands

- `define_dft opcg_domain` on page 725
- `define_dft opcg_mode` on page 728
- `define_dft opcg_trigger` on page 730
- `insert_dft opcg` on page 825

Sets these attributes: Osc Source Attributes
define_dft pmbist_direct_access

define_dft pmbist_direct_access -function string
-.active {low | high}
  (mbist_clock | port | pin)

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 Encounter Test 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.

- **-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 burnin patterns.

  - **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 burnin 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.

{mbist_clock | pin | port}

Specifies the source pin or 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 pin or port can be shared for functional uses provided the defined and applied PMBIST DFT configuration mode enables PMBIST usage during PMBIST operations.

Related Information

Affects this command: insert_dft pmbist on page 827
Sets these attributes: Programmable Direct Access Function Attributes
**define_dft preserved_segment**

```bash
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:

```bash
/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.

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).

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 buffer, an inverter, a (hierarchal) pin, or a scan segment.

-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.
  ```
  rc:/> define_dft preserved_segment -name segmenta -analyze \
  -sdi */u_a/SIa -sdo */u_a/Soa
  ```

- 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

See the following sections in Design for Test in Encounter RTL Compiler:

- Handling Preexisting Scan Segments
- Creating Head, Body, and Tail Segments
- Segmentation Rules in “Exporting the Design”

Affects this constraint: define_dft scan_chain on page 739

Affects these commands: connect_scan_chains on page 681
  
  report dft_chains on page 885

Sets these attributes: Scan Segment Attributes
**define_dft scan_chain**

```plaintext
define_dft scan_chain [ -name name ]
  [ -sdi sdi -sdo sdo [ -create_ports ]
    [ -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.
-configure_pad {tm_signal | se_signal}

Specifies the test signal (test mode or shift enable signal) that the RC-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.
Prevents that RTL Compiler reassociates (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.

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.

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.

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 RC-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.

Specifies the core-side hookup pin to be used for the scan data output signal during scan chain connection.
Note: When you specify this option, the RC-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_select test_signal

Specifies the select control signal to the mux inserted for a shared output port.
**Design for Test**

---

**Examples**

The following example creates a chain containing 3 segments previously defined:

```bash
rc:/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
```

---

**Note:** This option only applies to the muxed scan style.
The following example analyzes an existing scan chain:

```
rc:/> 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**

See the following sections in *Design for Test in Encounter RTL Compiler*

- Defining Scan Chains
- Creating Head, Body, and Tail Segments
- Analyzing Chains in a Scan-Connected Netlist

**DFT-Related Commands** in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affected by this constraint:  
- `define_dft abstract_segment` on page 689
- `define_dft fixed_segment` on page 704
- `define_dft floating_segment` on page 706
- `define_dft preserved_segment` on page 736
- `define_dft shift_enable` on page 751
- `define_dft shift_register_segment` on page 754
- `define_dft test_clock` on page 762

Affects these commands:  
- `connect_scan_chains` on page 681
- `report dft_chains` on page 885

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
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 signal for the design. If you specify more than one signal, the last definition overwrites the existing one.

The command returns the directory path to the test_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 RC-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
-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 RC-DFT engine does not validate the controlability 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.
Example

- The following example defines `sca` as the driver for the `scan_clock_a` signal and assigns `SCA` as the `test_signal` object name.

```plaintext
define_dft scan_clock_a -name SCA sca
```

Related Information

Defining LSSD Scan Clock Signals in Design for Test in Encounter RTL Compiler

Affects these commands:
- `connect_scan_chains` on page 681
- `report dft_chains` on page 885
- `write_atpg` on page 904

Sets these attributes:
- `Test Signal Attributes`

Affected by this attribute:
- `dft_scan_style`
**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 signal for the design. If you specify more than one signal, the last definition overwrites the existing one.

The command returns the directory path to the `test_signal` 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 RC-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
-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 RC-DFT engine does not validate the controllability 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.
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

**Defining LSSD Scan Clock Signals** in *Design for Test in Encounter RTL Compiler*

Affects these commands:
- `connect_scan_chains` on page 681
- `report dft_chains` on page 885
- `write_atpg` on page 904

Sets these attributes:
- `Test Signal Attributes`

Affected by this attribute:
- `dft_scan_style`

Note: If the test signal is marked as ideal, RTL Compiler 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 RC-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*
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 RC-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.
-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 RC-DFT engine does not validate the controllability 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, RTL Compiler 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 RC-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
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

- **Defining Shift-Enable Signals** in *Design for Test in Encounter RTL Compiler*
- **Rules for Constraining Test Signals** in *Interfacing between Encounter RTL Compiler and Encounter Conformal*

Affects these commands:

- `check_dft_rules` on page 648
- `connect_scan_chains` on page 681
- `insert_dft_shadow_logic` on page 846
- `report_dft_chains` on page 885

Sets these attributes:

- `Test Signal Attributes`
**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 RC-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 directory path to the object that it creates. 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.

**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

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Manually Identifying Shift Registers
- Creating Head, Body, and Tail Segments
- Identifying Shift Registers in a Mapped Netlist before Creating the Scan Chains

Affects this constraint:  `define_dft_scan_chain` on page 739
Affects these commands:  `connect_scan_chains` on page 681
                          `report_dft_chains` on page 885
Related command:  `identify_shift_register_scan_segments` on page 780
Sets these attributes:  *Scan Segment Attributes*
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.
-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**

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **Hookup Requirements for TAP Ports**
- **JTAG-Controlled Scan Modes**

Sets these attributes: **TAP Port Attributes**

- **dft_tap_tck_period**
**define_dft test_bus_port**

```plaintext
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] {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

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>capture_wr</td>
<td>WIR capture control</td>
</tr>
<tr>
<td>compress_sdi</td>
<td>Common scan data input (broadcast to multiple cores)</td>
</tr>
<tr>
<td>compress_sdo</td>
<td>Common scan data output</td>
</tr>
<tr>
<td>compression_clock</td>
<td>Test clock used for compression mask (and MISR if present)</td>
</tr>
<tr>
<td>compression_enable</td>
<td>Control to enable compression mode</td>
</tr>
<tr>
<td>custom</td>
<td>User specified control signal</td>
</tr>
<tr>
<td>mask_enable</td>
<td>Data input to enable channel masking</td>
</tr>
<tr>
<td>mask_load</td>
<td>Control to enable mask loading mode</td>
</tr>
</tbody>
</table>
misr_reset_enable
Control signal to reset the MISR (misr compression only)

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

shift_wr
WIR shift control

spread_enable
Control to enable the xor decompressor

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
Shared compression and wrapper clock

wrapper_clock
Wrapper instruction and dedicated wrapper cell clock

wrapper_reset
Wrapper instruction register reset control

**Note:** Some restrictions apply for the functions. Refer to *Design for Test in Encounter RTL Compiler* 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 RC-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 RC-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.

```
-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|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 \texttt{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 \texttt{-wir\_signal} option.

\textit{Default:} the value of the \texttt{-wir\_reset\_value} option

\section*{Related Information}

Hierarchical Test Flow: Preparing a Core in \textit{Design for Test in Encounter RTL Compiler}

Inserting Core-Wrapper Logic in \textit{Design for Test in Encounter RTL Compiler}

Affects this command: \texttt{insert\_test\_compression} on page 871

Sets these attributes: \textit{Test Bus Port Attributes}
**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]]`  
`[-controllable]`  
`{pin|port} [{pin|port}] ...`

Defines a test clock and associates a test 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.

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**

- **-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.

- **-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. 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

-hookup_pin pin
Specifies the core-side hookup pin to be used for the top-level test clock during DFT synthesis.
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
```

**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 Design for Test in Encounter RTL Compiler:

- 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 648
connect_scan_chains on page 681
fix_dft_violations on page 772
report_dft_chains on page 885

Sets these attributes: Test Clock Attributes
**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_in | -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 RC-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.
**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 RC-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, RTL Compiler 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`
-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 RC-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 not 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_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.

Note: This option cannot be specified with the -create_port option.
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.

-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

Defining Test Mode Signals in Design for Test in Encounter RTL Compiler

DFT-Related Commands in Interfacing between Encounter RTL Compiler and Encounter Conformal

Affects these commands: check_dft_rules on page 648

- fix_dft_violations on page 772
- insert_dft_shadow_logic on page 846
- insert_dft_test_point on page 852

Sets these attributes: Test Signal Attributes
dft_trace_back

dft_trace_back
  [-mode integer] [-polarity]
  [-continue] [-print]
  (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.

Options and Arguments

- **-mode integer** Specifies the mode for tracing back.
  - 0 does not perform constant propagation
  - 1 performs tied-constant propagation
  - 2 performs tied-constant and test-mode propagation
  - 3 performs tied-constant, test-mode and shift-enable propagation

  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)** 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.

Examples

- Following command traces back without performing constant propagation.

  rc:/> 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.

```
rc:/ 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.

```
rc:/ dft_trace_back /designs/top/instances_hier/g121/pins_in/CME -polarity 1 0
```
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 RTL Compiler 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.

  If you omit the design name, the top-level design of the current directory of the design hierarchy is used.
-\textbf{dont\_check\_dft\_rules}

Prevents the DFT rules from being checked automatically after fixing violations.

-\textbf{dont\_map}

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

-\textbf{exclude\_xsource\ instance}

Specifies instances to exclude from automatic fixing of X-source violations.

-\textbf{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.

\textbf{Note}: This option can only be used when fixing an asynchronous set reset violation and requires the -\textbf{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 -\textbf{test\_control} option.

To specify this option, the pin or port must first be declared as a \textbf{test\_mode} signal using the define\_dft \textbf{test\_mode} constraint.

-\textbf{preview}

Reports how the violations will be fixed, without making modifications to the netlist.

[-\textbf{rise} | -\textbf{fall}]

Specifies to use the rising or falling edge of the test clock to fix the DFT violation during test-mode operation.

\textbf{Default}: -\textbf{rise}

-\textbf{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.
This option is optional when fixing clock violations. By default, the RC-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 RTL Compiler to fix all clock, async set, and async reset violations using test mode signal tm and test clock CK1 using the rising edge as the active edge.

```shell
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,

```shell
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
```
In this case, the RC-DFT engine automatically determines which test clock to use to fix
the clock violations.

- The following example instructs RTL Compiler to fix all clock, async set, and async 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 RTL Compiler 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

Fixing DFT Rule Violations in Design for Test in Encounter RTL Compiler

Affected by these constraints: define_dft shift_enable on page 751

- define_dft test_clock on page 762
- define_dft test_mode on page 766

Affects these commands: check_dft_rules on page 648

- report dft_registers on page 888
- report dft_violations on page 890
- synthesize on page 377

Sets these attributes: dft_status

- dft_violation

Violations Attributes
**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 undergo analysis and to insert inverters.

**Related Information**

[Fixing Scan Path Inversions](#) in *Design for Test in Encounter RTL Compiler*
identify_domain_crossing_pins_for_cgic_and_scan_abstracts

identify_domain_crossing_pins_for_cgic_and_scan_abstracts
  [-cgic] [-bbox] [-add_fencing]
  [design]

Identifies domain crossing pins for blackboxes and integrated clock-gating instances in the specified design and adds blocking logic.

Options and Arguments

design
  Specifies the design in which you want to identify domain crossing pins.

-add_fencing
  Specifies to add fencing for the identified domain crossing pins

-bbox
  Specifies to identify domain crossing pins for blackboxes.

-cgic
  Specifies to identify domain crossing pins for integrated clock-gating instances.

Related Information

Inserting On-Product Clock Generation Logic in Design for Test in Encounter RTL Compiler

Related command: insert_dft_opcg on page 825
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.

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:

```bash
rc:/> identify_multibit_cell_abstract_scan_segments -preview
```

Would execute command:

```bash
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:

```bash
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:

```
rc:// 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
```

**Related Information**

**Mapping to Multi-Bit Scan Cells** in *Design for Test in Encounter RTL Compiler*
**identify_shift_register_scan_segments**

```bash
identify_shift_register_scan_segments
    [-min_length integer] [-max_length integer]
    [-preview] [-incremental]
```

Identifies all shift registers in the design whose minimum length either satisfies the default minimum length, or the specified minimum and maximum length values.

The RC-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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in *Design for Test in Encounter RTL Compiler*.

### Options and Arguments

- `-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.

- `-max_length integer`

  Specifies the maximum length that an automatically identified shift register can have.

  If the length of a functional shift register exceeds this length, it will be broken in multiple scan segments.
**Note:** There is no default for the maximum length.

**-min_length integer**

Specifies the minimum length a shift register must have to be automatically identified by the tool.

*Default:* 2

**-preview**

Reports which shift register segments will be identified, without adding them to the list of scan segments.

**Related Information**

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Automatically Identifying Shift Registers
- Identifying Shift Registers in a Mapped Netlist before Creating the Scan Chains
- Analyzing Chains in a Scan-Connected Netlist

Sets this attribute: user_defined_segment
identify_test_mode_registers

identify_test_mode_registers
  { -fixed_value_register_file file
    | {-stil file [-macro string] | -mode_init file)
    [-library string] [-et_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 RC-DFT engine.

The fixed-value registers can be identified

- From an existing ET log file
- By invoking Encounter Test and simulating a mode initialization sequence

Note: In the latter case, you need to have the Encounter Test software installed and your operating system PATH environment variable must include the path to the Encounter Test software. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

-continue_with_severe_warnings

Continues the Encounter Test 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.

-et_log string

Specifies the log file for Encounter Test.

-fixed_value_register_file file

Specifies an ET log file from a previous run that contains a list of fixed value registers.

When this option is specified, RTL Compiler can parse this file instead of invoking Encounter Test to get the list of fixed value registers.
Note: This option cannot be specified with the -stil, -mode_init, -macro, -library, -et_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 requests a report of the list of the registers with fixed values.

```
rc:/> 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
-----------------------------------------

<table>
<thead>
<tr>
<th>Test Function</th>
<th>Block Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+TI</td>
<td>tc/ts_reg[0]</td>
</tr>
<tr>
<td>+TI</td>
<td>tc/ts_reg[1]</td>
</tr>
<tr>
<td>+TI</td>
<td>tc/ts_reg[2]</td>
</tr>
</tbody>
</table>

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.

```bash
cr:/> 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

... 
```n
cr:/> 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

**Identifying Fixed-Value Registers** in *Design for Test in Encounter RTL Compiler*

Affected by these constraints:  
- `define_dft test_mode` on page 766
- `define_dft test_clock` on page 762

Sets this attribute:  
- `user_defined_signal`
insert_dft

insert_dft
  {  boundary_scan  |  compression_logic
    |  dfa_test_points  |  jtag_macro
    |  lockup_element  |  logic_bist  |  mbist  |  opcg
    |  ptam  |  rrfa_test_points  |  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.
dfa_test_points    Inserts test points based on Deterministic Fault Analysis.
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.
mbist               Inserts Memory Built-In-Self-Test (MBIST) logic to test targeted memories in 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.
rrfa_test_points   Inserts test points based on
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.

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 788  
insert_dft dfa_test_points on page 804  
insert_dft jtag_macro on page 808  
insert_dft lockup_element on page 812  
insert_dft logic_bist on page 813  
insert_dft mbist on page 819  
insert_dft opcg on page 825  
insert_dft pm bist on page 827  
insert_dft ptam on page 833  
insert_dft rrfa_test_points on page 836  
insert_dft scan_power_gating on page 843  
insert_dft shadow_logic on page 846  
insert_dft shift_register_test_points on page 851
insert_dft test_point on page 852
insert_dft user_test_point on page 858
insert_dft wrapper_cell on page 861
insert_dft wrapper_instruction_register on page 867
insert_dft wrapper_mode_decode_block on page 869
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]

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 Encounter Test license. For more information on
the exact product requirements, refer to Encounter Test Product Requirements for Advanced
Features in Design for Test in Encounter RTL Compiler.

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.
-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.
-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.

Refer to Pinmap File Format for more information.

**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.

-preview

Shows the potential changes, without making any modifications to 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.
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*

- **Inserting Boundary-Scan Logic**
- **Physical Boundary-Scan Insertion**
- **JTAG-Controlled Scan Modes**

**Related constraints:**
- `define_dft jtag_instruction_register` on page 712
- `define_dft jtag_macro` on page 714
- `define_dft shift_enable` on page 751
- `define_dft test_clock` on page 762
- `define_dft test_mode` on page 766

**Affects these commands:**
- `compress_scan_chains` on page 660
- `insert_dft mbist` on page 819

---

**-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:* `TCK`

---

**-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:* `TDI`

---

**-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`
insert_dft_ptam on page 833

Sets these attributes:

- **boundary_type**
- **dft_jtag_macro_exists**

**JTAG Port Attributes**
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]
    [-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]]
 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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

**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 Encounter Test.

  **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.

**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, `prefixcompression_enable`, where `prefix` is the value of the `dft_prefix` root attribute.
-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
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.
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.
-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.
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.
-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.
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.
  ```
  rc:/> define_dft scan_chain -name DFTChain1 -sdi SI1 -sdo SO1 -create_ports
  rc:/> define_dft scan_chain -name DFTChain2 -sdi SI2 -sdo SO2 -create_ports
  rc:/> define_dft scan_chain -name DFTChain3 -sdi SI3 -sdo SO3 -create_ports
  rc:/> define_dft scan_chain -name DFTChain4 -sdi SI4 -sdo SO4 -create_ports
  rc:/> define_dft scan_chain -name DFTChain5 -sdi SI5 -sdo SO5 -create_ports
  rc:/> llength [find / -actual_scan_chain *]
  55
  rc:/> insert_dft compression_logic -use_user_scan_chains \
  [find / -scan_chain DFTChain*] -auto_create
  rc:/> llength [find / -actual_scan_chain *]
  5
  ```
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **Low Pin Count Compression Using Encoded Compression Signals**
- **Inserting Compression Logic in a Design that Requires a Fixed Number of Channels**

Affected by these commands: `connect_scan_chains` on page 681

Affects this command: `report dft_chains` on page 885

Sets these attributes:
- `compressed`
- `dft_compression_signal`
- `dft_mask_clock`
- `dft_misr_clock`
- `type`
**insert_dft dfa_test_points**

insert_dft dfa_test_points -input_tp_file string
  [-max_number_of_testpoints integer]
  [-min_slack integer]
  [-fault_threshold integer]
  [-share_observation_flop integer]
  [-test_clock_pin {pin|port} ]
  [-test_control test_signal [-gate_clock]]
  [-gate_clock [-gate_clock_test_control test_signal]]
  [-control_only] [-observe_only]
  [-verbose] [design]

Inserts selected test points identified by Encounter Test Deterministic Fault Analysis. Test points can be inserted that have minimal impact on the slack and which target a minimum specified fault count.

**Tip**

The `insert_dft dfa_test_points` command is recommended to be run with the design in default timing mode. Ensure that the design is in default timing mode before running this command by running the following commands:

```
set default_mode [filter default true [find / -mode *]]  // to retrieve the default timing mode
report timing -mode $default_mode
```

**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 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.

- **-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**
  Enables the insertion of combinational logic to clock gate the test clocks of the inserted test points.
Note: If the hierarchy in which the observe points will be inserted already contains integrated clock-gating cells, the tool can use the ck_out pin of these existing clock-gating cells as the test clock source to be gated, if the test pin of the integrated clock-gating cell is connected to the driver of the test signal associated with the lp_clock_gating_test_signal attribute.

If the test pin of the integrated clock-gating cell is not controlled, the test synthesis (RC-DFT) engine will not identify the output pin of the integrated clock-gating cell as the local test clock source and instead will use the clock root itself as the test clock source to be gated to the clock pin of the observe test point.

Note: To use this command option you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

-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.

-input_tp_file file

Specifies the name of the file containing the test point locations. The file is specified in Encounter Test format.

-max_number_of_testpoints integer

Specifies the number of test points to be inserted.

If this option is not specified, then all the test points from the file specified with the -input_tp_file option will be processed for insertion.

Default: All

-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_observation_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.

**Examples**

- The following command inserts all of the DFA test points, and when possible 3 test point locations will be shared through an XOR-tree to a common observation flop.

  ```
  insert_dft dfa_test_points \
  -input_tp_file rc et dfa/myDesign.dfa.tpfile \ 
  -test_control test_mode \ 
  -test_clock_pin clk \ 
  -share_observation_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.

  ```
  insert_dft dfa_test_points -max_number_of_testpoints 500 \ 
  -fault_threshold 3 -test_control tm -test_clock_pin clk -input_tp_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.

  ```
  insert_dft dfa_test_points -min_slack 3000 -share_observation_flop 3 \ 
  -test_control tm -test_clock_pin clk -input_tp_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.

```
insert_dft dfa_test_points \
-input_tp_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_testpoints 10
```

The following command inserts a maximum of 10 test points for locations whose fault count is greater than or equal to 15:

```
insert_dft dfa_test_points \
-input_tp_file rc_et_dfa/myDesign.dfa.tpfile \ 
-test_control test_mode1 \ 
-test_clock_pin clk \ 
-max_number_of_testpoints 10 \ 
-fault_threshold 15
```

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Using Encounter Test to Perform a Deterministic Fault Analysis on a Scan Connected Netlist
- Inserting Scannable Test Points in Existing Scan Chains

Affected by these constraints:  
- `define_dft test_clock` on page 762
- `define_dft test_mode` on page 766
**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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in *Design for Test in Encounter RTL Compiler*.

**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 prefixtdi, prefixtdo, prefixtms, prefixtrst, and prefixtck, 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.
-dont_create_DFT_TDO_enable_port

Prevents the creation of the block-level prefixTDO_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, prefixTDO_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.
-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 Macro JTAG_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

-trrst 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
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 \ 
  -preserve_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

See the following sections in *Design for Test in Encounter RTL Compiler*:

■ Inserting a JTAG Macro

■ JTAG-Controlled Scan Modes

Related constraints: define_dft jtag_instruction on page 708

define_dft jtag_instruction_register on page 712

Affects these commands: compress_scan_chains on page 660

insert_dft mbist on page 819

insert_dft ptam on page 833

Sets these attributes: JTAG PORT Attributes

dft_jtag_macro_exists
**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 RC-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 RC-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**

- Analyzing Chains in a Scan-Connected Netlist in *Design for Test in Encounter RTL Compiler*

  Affected by this attribute: `dft_lockup_element_type`
insert_dft logic_bist

insert_dft logic_bist
  -bist_clock port [-bist_clock_hookup pin]
  [ -scan_clock port -test_control test_signal
    [ -scan_clock_hookup pin]
  | -dont_scan_lbist_flops]
[-inside instance] [-stagger_clocks]
[-add_buffer_to_bist_clock]
[-tester_available port...]
[-free_running_clocks test_clock...]
[-control_inputs_0 port...] [-control_inputs_1 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]
[-add_masking]
[-set_patterns integer] [-scan_pattern_counter_length integer]
[-static_patterns integer] [-static_pattern_counter_length integer]
[-dynamic_patterns integer] [-dynamic_pattern_counter_length integer]
[-scan_channel_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]
[-programmable_defaults] [design]

Inserts Logic Bist (LBIST) logic into the design.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

-add_buffer_to_bist_clock

Adds a buffer to the LBIST oscillator clock.

Specify this option when you use a top-level clock as LBIST oscillator clock to avoid having to manually specify the number of cycles required for LBIST when Encounter Test is used for simulation.
-add_masking

Inserts the LBIST macro including the optional channel masking logic (only available with SETBIST support).

-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.

-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_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.

-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_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
   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.

-dynamic_patterns integer
   Specifies the default number of dynamic set 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.

-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.
   By default, the LBIST logic is inserted as a hierarchical instance in the top-level of the design.
-**programmable_defaults**

Generates a parallel interface on the LBIST macro that can be used to configure LBIST parameters externally, even at run-time. Using this option, the RUNBIST and direct-access interfaces can achieve a level of control similar to the SETBIST interface, but without requiring a serial load of the parameters.

The generated interface allows setting all available counter and window values as well as PRPG and mask values.

When this option is omitted, these parameters are assumed to be constant and are hard-coded inside the macro, saving area but reducing flexibility.

-**reset_patterns integer**

Specifies the default number of reset test patterns to be executed.

-**reset_pattern_counter_length integer**

Specifies the length of the reset pattern counter that is part of the LBIST macro.

-**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_channel_counter_length integer**

Specifies the length of the scan channel counter that is part of the LBIST macro.

-**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.

-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
   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.

-stagger_clocks
   Inserts the LBIST macro with scan/capture clock staggering
   support.

-static_patterns integer
   Specifies the default number of static set 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.

-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.

-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

Inserting LBIST Logic in Design for Test in Encounter RTL Compiler

Affected by these constraints: define_dft_test_mode on page 766
Related commands: write_et_lbist on page 938
write_logic_bist_macro on page 954
insert_dft mbist

insert_dft mbist
    [-config_file config_file
        | -preview [-config_file config_file]]
    [-connect_to_jtag | -dont_create_mbist_ports
    |-direct_access_only [-dont_create_mbist_ports] ]
    [-dft_configuration_mode mode]
    [-test_control test_signal]
    [-dont_check_dft_rules]
    [-diagnose_mbist string] [-diagnose_rombist string]
    [-run_mbist string] [-read_mbist_string]
    [-continue_mbist string]
    [-design string] [-module_prefix string]
    [-directory dir_path] [-measure_ports ports]
    [-mode mode_name [-notmode mode_name...]]

Analyzes the design’s memories to generate a configuration file template when the -preview option is used, optionally specifying a configuration file containing memory module information using the -config_file option, or inserts Memory Built-In-Self-Test (MBIST) or MBIST supporting logic based on the definitions in the configuration file specified using the -config_file or -interface_file_dirs options.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

-config_file config_file

Specifies the file that contains the user-defined configuration parameters which selects the MBIST features and controls for the insertion of the test logic for targeted memories.

You can use this option in conjunction with the -interface_file_dirs option as part of a bottom-up flow. The -interface_file_dirs option points to files that describe the MBIST structures that have been inserted earlier in a block of this design.

You can use this option with the -preview option to supplement the library information for the memory modules in the design. In this situation, only module statements are permitted in the configuration file.
-connect_to_jtag  Connects MBIST logic's JTAG interface pins to a pre-instantiated JTAG macro (either Cadence's or Third party).

Use this option when the instance is the top level of the chip, and you have instantiated the JTAG macro and will use it to access the MBIST engines.

-continue_mbist  

Specifies the CONTINUE_MBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.

Default: CONTINUE_MBIST

-design design  Specifies the name of the top-level design.

-dft_configuration_mode  

Specifies the configuration mode in which MBIST will be operating. MBIST test pin verification and netlist back-tracing using the check_mbist_rules command must be done in this mode as well.

Default: mbist

-diagnose_mbist  

Specifies the DIAGNOSE_MBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.

Default: DIAGNOSE_MBIST

-diagnose_rombist  

Specifies the DIAGNOSE_ROMBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.

Default: DIAGNOSE_ROMBIST

-direct_access_only  

Inserts MBIST logic and makes the necessary connections using the MBIST direct access information only. In this case, any information for the JTAG macro is ignored even when a JTAG macro is specified.

-directory directory_path  

Specifies the directory to which the interface files (MBIST pattern control and TDR mapping files) must be written. In a bottom-up flow, this directory is specified as input using the -interface_file_dirs option, when this command is run on a higher level of the hierarchy.
Default: \texttt{current\_working\_directory/mbist\_design}

-\texttt{dont\_check\_dft\_rules}

Prevents the DFT rules from being automatically checked after MBIST insertion.

-\texttt{dont\_check\_mbist\_rules}

Prevents the MBIST rules from being automatically checked after MBIST insertion.

-\texttt{dont\_create\_mbist\_ports}

Prevents creation of MBIST control ports for subsequent JTAG macro and direct access connection at the top-level instance specified by the \texttt{-instance} option.

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.

-\texttt{dont\_map}

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

-\texttt{interface\_file\_dirs string}

Specifies a list of interface file directories which contain the MBIST pattern control and TDR mapping files for blocks in which MBIST logic has already been inserted. In a bottom-up flow, these files represent an abstract model of the BIST-ed blocks and are used by the \texttt{insert\_dft mbist} command when processing the larger design. Separate the directory names with blank spaces.

-\texttt{measure\_ports ports}

Specifies the tester-accessible ports to be used as bitmap pattern measure ports in addition to the default JTAG TDO port.

\textbf{Note:} This option applies to MBIST bitmap patterns and chip-level designs only.

-\texttt{mode mode\_name}

Specifies the name of the timing mode for which you want to perform timing analysis on the MBIST logic only. The command adds timing constraints for the specified mode.
Do not use this option, if you want to perform timing analysis on MBIST in functional timing mode with the rest of the functional design.

**Note:** The specified mode should exist in the `/designs/design/modes` directory.

```
-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 mbist` command.

**Note:** This option is required in case of an MBIST block-level insertion flow.

```
-notmode mode_name
```

Specifies the name(s) of the timing mode(s) in which the MBIST logic should not be considered during timing analysis. You can only specify this option if you specified the `-mode` option.

**Note:** The specified mode should exist in the `/designs/design/modes` directory.

```
-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.

This option can be used with the `-config_file` option to supplement the library information for the memory modules in the design. In this situation, only module statements are permitted in the configuration file.

```
-read_mbist string
```

Specifies the `READ_MBIST` instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.

*Default:* `READ_MBIST`

```
-run_mbist string
```

Specifies the `RUN_MBIST` instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.

*Default:* `RUN_MBIST`

```
-test_control test_signal
```

Indicates the user-defined test-control signal which must be held in its inactive state when the chip is in MBIST mode.

This signal will be asserted when the chip is in ATPG/SCAN mode to test MBIST logic for manufacturing defects.
Examples

- The following command analyzes the netlist to identify memory instances and generates an MBIST configuration file template for this design.
  
  `insert_dft mbist -preview`

- The following command analyzes the netlist to identify memory instances and generates an MBIST configuration file template for this design with guidance on memory characteristics from the user-specified configuration file.
  
  `insert_dft mbist -preview -config_file \ ..;/et_Inputs/my_module_configurations.txt`

- The following command inserts the MBIST logic as described in the configuration file, by default, to the design module at the highest level of hierarchy of the design.
  
  `insert_dft mbist -config_file ..;/et_inputs/my_configuration.txt`

- The following command inserts the MBIST logic as described in the configuration file, into the design module `chip_top` of the design.
  
  `insert_dft mbist -config_file ./et_inputs/my_configuration.txt
   -instance chip_top`

- The following command inserts the MBIST logic as described in the configuration file, into design module `chip_top` placing interface files into a specified directory.
  
  `insert_dft mbist -config_file ./et_inputs/my_configuration.txt
   -instance chip_top -directory ./my_et_files`

- The following command inserts the MBIST logic as described in the configuration file, into the design module `chip_top` of the design. The active high `ScanTestMode` signal will be deasserted during MBIST test operations.
  
  `define_dft test_mode -name ScanTestMode -active high \ [find ./des* -pin ScanTestMode]
   insert_dft mbist -config_file ./et_inputs/my_configuration.txt
   -instance chip_top -test_control ScanTestMode`

Related Information

- _Design Flows_ in “Inserting Memory Built-In-Self-Test Logic” in _Design for Test in Encounter RTL Compiler_

  Affected by these constraints:  
  
  - `define_dft dft_configuration_mode` on page 699
  - `define_dft mbist_clock` on page 719
  - `define_dft mbist_direct_access` on page 722
  - `define_dft test_mode` on page 766
Affects these commands:

- `insert_dft boundary_scan` on page 788
- `insert_dft jtag_macro` on page 808
- `write_do_lec` on page 257
- `write_sdc` on page 292

Related commands:

- `check_mbist_rules` on page 654
- `write_dft_rtl_model` on page 921
- `write_et_bsv` on page 930
- `write_et_mbist` on page 942
- `write_mbist_testbench` on page 958

Related attributes:

- `dft_rtl_insertion`
- `mbist_instruction_set`
insert_dft opcg

insert_dft opcg
   -opcg_enable test_signal
   -opcg_load_clock test_clock
   -scan_clock test_clock
   -test_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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.

-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.

\[-test_enable \text{ test_signal}\]

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

\[
\text{insert_dft opcg -opcg_enable OPCGMODE -opcg_load_clock opcg_load_clk \}
\text{ -scan_clock scanclk -test_enable testmode}
\]

Related Information

Inserting the OPCG Logic in Design for Test in Encounter RTL Compiler

Related constraints: define_dft shift_enable on page 751

define_dft test_clock on page 762

define_dft test_mode on page 766
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]
  [-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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

**Options and Arguments**

- **-alt_dft_configuration_mode mode**
  Specifies the configuration mode in which PMBIST will be operating for non-JTAG mode of operation. PMBIST test pin verification must be done in this mode as well.

  **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
-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
-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 instantiatted 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 no test signals must 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
-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
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

Design Flows’ in Inserting Programmable Memory Built-In-Self-Test Logic in Design for Test in Encounter RTL Compiler

Affected by these constraints:

- define_dft dft_configuration_mode on page 699
- define_dft mbist_clock on page 719
- define_dft pmbist_direct_access on page 734
- define_dft test_mode on page 766

Affects these commands:

- insert_dft boundary_scan on page 788
- insert_dft jtag_macro on page 808

Related commands:

- read_memory_view on page 879
- read_pmbist_interface_files on page 881
- write_pmbist_interface_files on page 962
- write_pmbist_testbench on page 964

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
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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.
-`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.

  ```bash
  insert dft ptam -instruction PTAM -power_test_enable /chip_top/PwrTe \ -directory ${workdir}/testmode_data
  ```
Related Information

Inserting Power Test Access Mechanism (PTAM) Logic in Design for Test in Encounter RTL Compiler

Affected by these commands: read_power_intent on page 1032
define_dft_shift_enable on page 751
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
insert_dft rrfa_test_points

insert_dft rrfa_test_points
{ -input_tp_file file
  | [-atpg [-atpg_options string]
  [-build_model_options string]
  [-build_faultmodel_options string]
  [-build_testmode_options string]]
  [-rrfa_effort {low|medium|high}]
  [-rrfa_options string]
  -directory string [-et_log file] [-verbose]
  [-output_tp_file file] )
  [-max_number_of_testpoints integer ]
  [-min_slack integer]
  [-share_observation_flop integer]}
  [-test_clock_pin {port|pin}]
  [-test_control test_signal]
  [-gate_clock [-gate_clock_test_control test_signal]]
  [-control_only] [-observe_only]
  [-library string] [design]

Invokes Encounter Test to

■ Perform Automatic Test Pattern Generator (ATPG) based testability analysis to prune out the ATPG detectable faults (if the -atpg option is selected)

■ Choosing the -atpg option does affect the runtime.

■ Perform Random Resistance Fault Analysis (RRFA) based testability analysis and test-point selection

■ Insert selected test points that have minimal impact on the slack

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Tip

The insert_dft rrfa_test_points command is recommended to be run with the design in default timing mode. Ensure that the design is in default timing mode before running this command by running the following commands:

```
set default_mode [filter default true [find / -mode *]]  // to retrieve the default timing mode

report timing -mode $default_mode
```
Options and Arguments

-atpg

Runs ATPG-based testability analysis to prune the ATPG detectable faults before running random-resistant fault analysis.

-atpg_options string

Specifies extra options to run ATPG-based testability analysis in a string.

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 Encounter Test documentation).

Note: This option is mutually exclusive with the -input_tp_file option.

-build_model_options {option1=value option2=value}

Specifies extra options to apply when building the Encounter Test model.

Note: For more information on these options, refer to the build_model command in the Command Line Reference (of the Encounter Test documentation).

-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 Encounter Test documentation).

-build_testmode_options {option1=value option2=value}

Specifies extra options to apply when building the test mode for Encounter Test.

Note: For more information on these options, refer to the build_testmode command in the Command Line Reference (of the Encounter Test documentation).

-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 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.

\textbf{-directory} \textit{string}  
Specifies the working directory for Encounter Test.

\textbf{Note:} This option is only required when you run an RRFA-based analysis

\textbf{-et\_log} \textit{file}  
Specifies the name of the Encounter Test log file. This file will be generated in the specified directory.

\textbf{Default:} \textit{eta\_from\_rc.log}

\textbf{-gate\_clock}  
Enables the insertion of combinational logic to clock gate the test clocks of the inserted test points.

\textbf{Note:} If the hierarchy in which the observe points will be inserted already contains integrated clock-gating cells, the tool can use the \textit{ck\_out} pin of these existing clock-gating cells as the test clock source to be gated, if the test pin of the integrated clock-gating cell is connected to the driver of the test signal associated with the \textit{lp\_clock\_gating\_test\_signal} attribute.

If the test pin of the integrated clock-gating cell is not controlled, the test synthesis (RC-DFT) engine will not identify the output pin of the integrated clock-gating cell as the local test clock source and instead will use the clock root itself as the test clock source to be gated to the clock pin of the observe test point.

\textbf{Note:} To use this command option you need an Encounter Test license. For more information on the exact product requirements, refer to \textit{Encounter Test Product Requirements for Advanced Features} in \textit{Design for Test in Encounter RTL Compiler}.

\textbf{-gate\_clock\_test\_control} \textit{test\_signal}  
Specifies the test signal to be used to control the gating logic of the clock of the test point.

\textbf{-input\_tp\_file} \textit{file}  
Specifies the name of the file containing the test point locations. The file is specified in Encounter Test format.

If you do not specify this option, the test point locations are read from
The file specified with the -output_tp_file option if you also specified the -rrfa option

The following file in the working directory if you did not specify any file:

TB/testresults/TestPointInsertion.ASSUMESCAN.expt.

Note: This option is mutually exclusive with -atpg_options and -rrfa_options.

-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:

   insert_dft rrfa_test_points -library "./padcells.v \
   ./stdcells.v ./memories/*.v \
   ./ip_blocks/*.v"

2. Using an include file.

   insert_dft rrfa_test_points \
   -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: This option is only required when you invoke this command on a mapped netlist.

-max_number_of_testpoints 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_tp_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.

-output_tp_file file  Specifies the output file generated by the RRFA-based analysis. 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.

-rrfa_effort {low | medium | high}  Specifies the effort to be used for the RRFA-based analysis.

  Default: low

-rrfa_options string  Specifies the extra options to run RRFA-based testability analysis in a string.

  For more information on these options, refer to the analyze_random_resistance command in the Command Line Reference (of the Encounter Test documentation).

  Note: This option is mutually exclusive with the -input_tp_file option.

-share_observation_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.

**Examples**

- The following example performs only RRFA-based testability analysis and creates a file `myfile` with the suggested test point locations.

  ```plaintext
  insert_dft rrfa_test_points -output_tp_file myfile
  ```

- The following example inserts 10 test points from the specified test point file `myfile`.

  ```plaintext
  insert_dft rrfa_test_points -max_number_of_testpoints 10 \
  -test_control tm -test_clock_pin clk -input_tp_file myfile
  ```

- The following example performs ATPG-based testability analysis followed by RRFA-based testability analysis. The command generates a report on the fault coverage in the log file, and stores the suggested test point locations in the `TB/testresults/TestPointInsertion.ASSUMESCAN.expt` file.

  ```plaintext
  insert_dft rrfa_test_points -atpg
  ```

- The following example performs ATPG-based testability analysis, RRFA-based testability analysis, and inserts 10 test points. During RRFA-based testability analysis, test point locations are written to the `TB/testresults/TestPointInsertion.ASSUMESCAN.expt` file, while during test point insertion, they are read from this file.

  ```plaintext
  insert_dft rrfa_test_points -atpg -max_number_of_testpoints 10 \
  -test_control tm -test_clock_pin clk
  ```

**Related Information**

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **Using Encounter Test to Automatically Select and Insert Test Points**

- **Requirements** in “Inserting Logic Built-In-Self-Test Logic”
Inserting Scannable Test Points in Existing Scan Chains

Affected by these constraints:  

define_dft_test_mode on page 766  
define_dft_test_clock on page 762
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.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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 Encounter Test 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.
-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.ASSUMESCANE.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.
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 Design for Test in Encounter RTL Compiler.

Affected by these constraints: define_dft shift_enable on page 751
define_dft test_mode on page 766
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 unit-directional pins only. Bidirectional pins and pins associated with test clock objects are skipped.

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 ... output_j...}
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

-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:

```
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 \( i_1 \) to output \( o_1 \), and input \( i_2 \) to output \( o_3 \) (skipping \( o_2 \)). The second command pairs input \( i_1 \) and output \( o_2 \).

```bash
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.

```bash
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.

```bash
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, \( i_1 \) and \( o_2 \) share a test point, and \( i_2 \) and \( o_1 \). In addition, no control point is inserted for the net driven by \( U_1/o_3 \).

```bash
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.

```bash
rc:/> 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

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

Inserting DFT Shadow Logic in *Design for Test in Encounter RTL Compiler*

Affects these commands:
- `check_dft_rules` on page 648
- `report_dft_registers` on page 888

Related constraints:
- `define_dft_shift_enable` on page 751
- `define_dft_test_clock` on page 762
- `define_dft_test_mode` on page 766
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

Inserting Test Points to Create Shift Registers in a Mapped Netlist before Creating the Scan Chains in Design for Test in Encounter RTL Compiler

Related constraint: define_dft shift_enable on page 751
insert_dft test_point

insert_dft test_point -location {pin|port}...
   [-test_control test_signal [-gate_clock]]
   -type { control_0 | control_1 } |
      {async_0 | async_1 | async_any
       | control_node -node {pin|port}
       | control_observe_0 | control_observe_1
       | control_observe_node -node {pin|port}
       | control_scan
       | observe_scan [-max_observe_share integer]
       | scan | sync_0 | sync_1 | sync_any }
   -test_clock_pin {pin|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 in 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**

-gate_clock

Enables the insertion of combinational logic to clock gate the test clocks of the inserted test points.
Note: If the hierarchy in which the observe points will be inserted already contains integrated clock-gating cells, the tool can use the ck_out pin of these existing clock-gating cells as the test clock source to be gated, if the test pin of the integrated clock-gating cell is connected to the driver of the test signal associated with the lp_clock_gating_test_signal attribute.

If the test pin of the integrated clock-gating cell is not controlled, the test synthesis (RC-DFT) engine will not identify the output pin of the integrated clock-gating cell as the local test clock source and instead will use the clock root itself as the test clock source to be gated to the clock pin of the observe test point.

Note: To use this command option you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

-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 856.

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 or control_observe_node 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.
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**.

**-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_dft shift_enable** or **define_dft 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.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control_observe_0</td>
<td>Inserts a control and an observation point. The control point is forced to the value 0.</td>
</tr>
<tr>
<td>control_observe_1</td>
<td>Inserts a control and an observation point. The control point is forced to the value 1.</td>
</tr>
<tr>
<td>control_observe_node</td>
<td>Inserts a control and an observation point. The control point is forced to the value of the node specified by -node.</td>
</tr>
<tr>
<td>control_scan</td>
<td>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.</td>
</tr>
<tr>
<td>observe_scan</td>
<td>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.</td>
</tr>
<tr>
<td>scan</td>
<td>Inserts a scannable control and observation test point.</td>
</tr>
<tr>
<td>sync_0</td>
<td>Inserts a synchronous control test point that forces the control point to the value 0.</td>
</tr>
<tr>
<td>sync_1</td>
<td>Inserts a synchronous control test point that forces the control point to the value 1.</td>
</tr>
</tbody>
</table>

**Note:** This option requires you to specify the `-test_clock_pin` option.
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`

Related Information

Inserting a Control and Observation Test Point in Design for Test in Encounter RTL Compiler.

Affects these commands:

- `check_dft_rules` on page 648
- `connect_scan_chains` on page 681
- `synthesize` on page 377
Related constraints:  
define_dft shift_enable on page 751  
define_dft test_clock on page 762  
define_dft test_mode on page 766  

Related attributes:  
Test Clock Attributes  
Test Signal Attributes
insert_dft user_test_point

Syntax:
```
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.
- **-cfo {pin | port}**: Specifies the cell functional output (CFO) pin or port name.
- **-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**: 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:

    rc:/> 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:

    rc:/> 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 Design for Test in Encounter RTL Compiler:

- 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 648
                              connect_scan_chains on page 681
                              synthesize on page 377
insert_dft wir_signal_bits

insert_dft wir_signal_bits
  [-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.

Options and Arguments

  -design design Specifies the design in which you want to insert the wrapper instruction register.
    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 Design for Test in Encounter RTL Compiler

Related command: define_dft test_bus_port on page 758
insert_dft wrapper_cell

insert_dft wrapper_cell -location pin_list
    [-floating_location_ok]
    [-skipped_locations_variable Tcl_variable]
    [-shared_through {buffer|inversers_and_buffers|combinational}]
      [-no_mux_before_shared_cell]]
    [-wck pin] -wsen_in pin -wsen_out pin
    [-decoded_select_cfi pin]
    [-wint (pin|port|constant)]
    [-wext (pin|port|constant) [-wcap pin]]
    [-guard {0|1} -wig pin -wog pin]
    [-dont_reuse_input_wrappers_for_output_ports]
    [-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}...]
    [-respect_dft_constants {test_setup | tied}]
    [-wrap_tied_constant_ports] [-inside_core]
    [-map_to_mux_for_dedicated_wrapper]
    [-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.

Options and Arguments

-decoded_select_cfi pin

Specifies the source pin name of the decoded control logic for the select-cfi signal.
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.

`-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.

`-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.

`-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_shared_threshold integer` Specifies the maximum number of scanable flops that can be shared in wrapper cells. When the number of scanable 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 dcw_cfi_mux and dcw_guard_mux respectively.

  **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_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.

-override_threshold_use_shared {pin|port}...

Specifies the list of pins or ports 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.

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.

\(-\text{skipped\_locations\_variable TCL\_variable}\)

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.

\(-\text{wcap driver}\)

Specifies the capture control source pin or port name.

Note: This option is ignored if you specified the \(-\text{decoded\_select\_cfi}\) option.

\(-\text{wck driver}\)

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.

\(-\text{wig driver}\)

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 \(-\text{wext}\) control signal.

\(-\text{wint \{pin|port|constant\}}\)

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 \(-\text{decoded\_select\_cfi}\) option.
-wext \{pin|port|constant\}

Specifies the control source pin or port name, or constant value for outward facing test mode.

**Note:** This option is ignored if you specified the -decoded_select_cfi option.

-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.

**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" 
  ```

**Related Information**

**Inserting the Wrapper Cells** in *Design for Test in Encounter RTL Compiler*

Affects this command: [connect_scan_chains](#) on page 681

Sets this attribute: [core_wrapper](#)
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.
-wrstn test_signal
   Specifies the test signal used as reset signal for the Wrapper Instruction Register.

-wsi port
-wso port
   Specifies the port used as Wrapper Serial Input.
   Specifies the port used as Wrapper Serial Output.

Related Information

Affected by this command: define_dft test_bus_port on page 758
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.

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
   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.

```bash
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 inttest 
  -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 inttest -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 TS2 TS1
define_dft dft_configuration_mode -name parallelCompress -type wrapper 
  -usage inttest -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

Inserting the Wrapper Mode Decoder in Design for Test in Encounter RTL Compiler

Affects these commands:  
  concat_scan_chains on page 675
  connect_scan_chains on page 681

Affected by this command:  
  define_dft dft_configuration_mode on page 699
insert_test_compression

insert_test_compression
   [-use_existing_channels actual_scan_chain...]
   | -build_new_scan_chains integer
   [-asymmetric_scan_ins integer]
   [-decompressor {broadcast | xor }]
   [ -compressor xor [-mask {wide1|wide2}]
   [ -scan_in_pipeline_depth integer]
   [-scan_out_pipeline_depth integer]
   | -compressor misr [-mask {wide0|wide1|wide2}]
   | -compressor hybrid [-mask {wide0|wide1|wide2}]
   [ -use_existing_wrapper_channels actual_scan_chain...]
   [ -auto_create] [-bypass_reg] [-use_wir_macro instance [-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.

   For the names of the auto-created ports, refer to Hierarchical Test.

   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.

-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

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.

-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.

-scan_in_pipeline_depth integer

Specifies the number of pipeline stages required at the scan data input side.

This option can only be specified for an XOR-based compressor.
-scan_out_pipeline_depth integer
   Specifies the number of pipeline stages required at the scan data output side.
   This option can only be specified for an XOR-based compressor.

-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

Hierarchical Test Flow: Preparing a Core in Design for Test in Encounter RTL Compiler

Related command: define_dft test_bus_port on page 758
map_mbist_cgc_to_cgic

map_mbist_cgc_to_cgic
    [-design design]
    [-clock_gating_cell libcell]

Maps the clock-gating logic inserted by the MBIST application to the specified integrated clock-gating cell.

Use this command if mapping was prevented during MBIST insertion.

Options and Arguments

-clock_gating_cell libcell
    Specifies the name of a clock-gating cell whose clock_gating_integrated_cell libcell attribute value should equal latch_posedge_precontrol.

    **Note:** If you omit the -clock_gating_cell option, the tool checks if the lp_clock_gating_cell attribute is specified on the module containing the clock-gating logic. If this libcell has the correct type, it will be used to replace the clock-gating logic, otherwise the tool will try to find the proper type of libcell inside the appropriate library domain to do the mapping. If you specify the wrong clock-gating cell type, a warning message will be issued and no clock-gating logic will be replaced.

-design design
    Specifies the name of the top-level design.

Related Information

**Mapping Clock-Gating Logic Inserted by insert_dft mbist to Clock-Gating Integrated Cells in Design for Test in Encounter RTL Compiler**

Related command: insert_dft mbist on page 819
read_dft_abstract_model

read_dft_abstract_model
   [-ctl [-use_scan_structures_se_only]
   [-override_scan_libcell]]
   [-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

-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.

   **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.

-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.

Note: This option must be specified with the -ctl option.
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Set Up for DFT Rule Checker
- Hierarchical Compression Flow
- Bottom-Up Test Synthesis Flow
- Analyzing Chains in a Scan-Connected Netlist

Affected by these commands:  
- `check_dft_rules` on page 648
- `connect_scan_chains` on page 681
- `synthesize` on page 377

Related commands:  
- `write_dft_abstract_model` on page 918
- `write_hdl` on page 276 (-abstract)

Sets these attributes:  
- **Scan Segment Attributes**
**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**

**Reading an IOSpecList File** in *Design for Test in Encounter RTL Compiler*

Affects these commands: `insert_dft boundary_scan` on page 788
`write_io_speclist` on page 952
read_memory_view

read_memory_view
-{-cdns_memory_view_file file
 | -preview [-directory string] }
[design]

Loads the configuration view file 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. After the configuration view file is loaded, RTL Compiler updates the design hierarchy with the views information and prints summary tables.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

### Options and Arguments

- **-cdns_memory_view_file config_file**
  Specifies the file containing the physical view of the memories in Cadence format. This file can contain information that is missing in the .lib files.

- **design**
  Specifies the name of the top-level design.

- **-directory string**
  Specifies the directory where the template file must be generated.

  **Default:** ./pmbist_design

- **-preview**
  Requests the generation of a template file for the configuration view file.

### Related Information

**Affects this command:**
insert_dft_pmbist on page 827

**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
read_pmbist_interface_files

read_pmbist_interface_files
   -directory string (design|subdesign)

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 Encounter Test create_embedded_test command.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

{design|subdesign}

Specifies the name of the design or subdesign for which the interface files are read.

(directory string

Specifies the directory that contains the interface files.

Related Information

Affects this command: insert_dft_pmbist on page 827

Related commands: write_pmbist_interface_files on page 962
replace_opcg_scan

replace_opcg_scan
   -edge_mode test_signal
   [-dont_map] [-effort {high | low}] [design]

Replaces domain blocking scan flops with their OPCG-equivalent flops.

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
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*

- Inserting the Toggle Muxes
- Top-Down Test Synthesis Flow with OPCG Logic Insertion
- Requirements in “Inserting Memory Built-In-Self-Test Logic”

Affected by these commands:  

- `reset_opcg_equivalent` on page 894
- `set_opcg_equivalent` on page 898
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.

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

See the following sections in Design for Test in Encounter RTL Compiler

■ Defining Scan-Equivalency between Non-Scan and Scan Flops to Map to Scan
■ Controlling Mapping to Scan in a Mapped Netlist
■ Identifying Shift Registers in a Mapped Netlist before Creating the Scan Chains

Affected by these commands: check_dft_rules on page 648
    identify_test_mode_registers on page 782
    set_scan_equivalent on page 900

Affected by this attribute: dft_scan_map_mode
report dft_chains

Refer to report dft_chains in the Chapter 9, “Analysis and Report.”
report dft_clock_domain_info

Refer to `report dft_clock_domain_info` in the Chapter 9, “Analysis and Report.”
report dft_core_wrapper

Refer to report dft_core_wrapper in the Chapter 9, “Analysis and Report.”
report dft_registers

Refer to `report dft_registers` in the Chapter 9, “Analysis and Report.”
report dft_setup

Refer to report dft_setup in Chapter 9, “Analysis and Report.”
report dft_violations

Refer to report dft_violations in Chapter 9, “Analysis and Report.”
report opcg_equivalents

Refer to report opcg_equivalents in Chapter 9, “Analysis and Report.”
report scan_compressibility

Refer to report scan_compressibility in Chapter 9, “Analysis and Report.”
report test_power

Refer to report test_power in the Chapter 9, “Analysis and Report.”
reset_opcg_equivalent

reset_opcg_equivalent [libcell]...

Removes the specified scan library cells from the OPCG-equivalency 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-equivalency table.

Example

The following example removes the snl_ffqx1 cell from the OPCG-equivalency table.
reset_opcg_equivalent snl_ffqx1

Related Information

Affects this command: replace_opcg_scan on page 882
Related command: set_opcg_equivalent on page 898
reset_scan_equivalent

reset_scan_equivalent [libcell]...

Removes the specified non-scan library cells from the scan-equivalency 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-equivalency table.

Example

The following example removes the snl_ffqxl cell from the scan-equivalency table.
reset_scan_equivalent snl_ffqxl

Related Information

Affects this command: replace_scan on page 884
Related command: set_scan_equivalent on page 900
set_compatible_test_clocks

```plaintext
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.
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Mixing Different Test Clocks in the Same Scan Chain
- Concatenating Scan Chains

Affects this command: `connect_scan_chains` on page 681
Related attribute: `dft_mix_clock_edges_in_scan_chains`
**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-equivalency 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.
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 SDFQ_X1M -opcg_cell S2DFQQN_X1M \
-edge_mode_pin TEL -loop_back_pin TI
```

Related Information

**Inserting the Toggle Muxes** in *Design for Test in Encounter RTL Compiler*

- Affects this command: [replace_opcg_scan](#) on page 882
- Related command: [reset_opcg_equivalent](#) on page 894
set_scan_equivalent

\[ \text{set_scan_equivalent} \]

- \text{-non_scan_cell libcell} \text{-scan_cell libcell} \]

\[-\text{tieoff_pins string}] \[-\text{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 \text{replace_scan} command to perform the actual conversion to scan.

\textbf{Note:} The RC-DFT engine automatically derives the scan data input, scan data output, and other test signals from the \text{test_cell} description of the scan flop in the target library.

\textbf{Options and Arguments}

- \text{-non_scan_cell libcell} Specifies a non-scan flip-flop library cell.

- \text{-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 \text{list_of_pin_groups} has the following format:

  \[
  \{(\text{non\_scan\_pin scan\_pin}) \{(\text{non\_scan\_pin scan\_pin})\ldots
  \]

- \text{-scan_cell libcell} Specifies a scan flip-flop library cell.

- \text{-tieoff_pins string} Specifies the tie-off value for extra scan cell pins.

  The \text{string} has the following format:

  \[
  \{(\text{pin value}) \{(\text{pin value}) \ldots
  \]

  The value can be a logic \text{0} or \text{1}.

\textbf{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.

\[ \text{set_scan_equivalent} \text{-non_scan_cell snl\_ffqxl} \text{-scan_cell snl\_sffqxl} \]
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Defining Scan-Equivalency between Non-Scan and Scan Flops to Map to Scan
- Controlling Mapping to Scan in a Mapped Netlist

Affects this command: `replace_scan` on page 884
Related command: `reset_scan_equivalent` on page 895
update_scan_chains

update_scan_chains
    [-flops instances] [-chains actual_scan_chains]
    [-type {dfa|rrfa|user}] [-max_print_flops integer]
    [-preview] [design] [> file]

Includes the identified scannable test points in the existing scan chains. These test points were inserted in the design after the scan chains were connected.

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.

-max_print_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}
    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.
    By default, all test point types are inserted.

Example

The following command requests 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

Inserting Scannable Test Points in Existing Scan Chains in Design for Test in Encounter RTL Compiler

Affected by these commands:
- `insert_dft dfa_test_points` on page 804
- `insert_dft rrfa_test_points` on page 836
- `insert_dft test_point` on page 852
- `insert_dft user_test_point` on page 858

Related command: `connect_scan_chains` on page 681
write_atpg

write_atpg
    { -cadence [-compression | > file]
      | -mentor [> file]
      | -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 Encounter Test ATPG tool.
    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.
-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

Creating an Interface File for ATPG Tool in Design for Test in Encounter RTL Compiler

Affected by this command: compress_scan_chains on page 660
connect_scan_chains on page 681
define_dft scan_clock_a on page 745
define_dft scan_clock_b on page 748
define_dft test_clock on page 762

Affected by these attributes: Actual Scan Chain Attributes
write_bsdll

write_bsdll
    [-pinmap_file file]
    [-bsdl_package_name files]
    [-bsdlout file]
    [-include_private_instructions]
    [-expose_ports_with_pinmap]
    -directory string

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.

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.bsdll.

-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:

```plaintext
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.

  Refer to Pinmap File Format for more information.

### 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_bsd1 -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 ;
  ```
Related Information

Writing a BSDL File in Design for Test in Encounter RTL Compiler

Affected by this command: insert_dft_boundary_scan on page 788
Related constraints: define_dft_shift_enable on page 751
define_dft_test_clock on page 762
define_dft_test_mode on page 766
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 Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

**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.

  For wide1 type of masking, the number of scan data input pins must equal the number specified for the -chains option minus one (\(\text{number\_of\_chains}-1\)), while for wide2 type of masking, the number must equal the specified number of chains minus two (\(\text{number\_of\_chains}-2\)).
-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.

-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. This script can be sourced into the current session to provide more information about the compression macro to commands such as write_et so they can generate accurate input files for Encounter Test.

-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.

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

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.
-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 Encounter Test 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.

  rc:/> write_compression_macro -compressor xor -chains 8 -sub_chains 88 \ -file xor1.v

  Checking out license ‘Encounter_Test_Architect’... (1 seconds elapsed)

- 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.

rc:/> 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.

rc:/> 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.

rc:/> 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 -decompressof 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.

write_compression_macro -chains 2 -sub_chains 32 -mask wide1 \  
-compressor smartScan_xor -decompressof xor -smartscan_ratio 4 \  
-smartscan_serial_only
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- **Low Pin Count Compression Using Encoded Compression Signals**
- **Manually Inserting a Scan Compression Macro**
- **Using Asymmetrical Scan Compression**
- **MxN Compression**
write_dft_abstract_model

write_dft_abstract_model [-ctl]
 [-include_compression_information]  
 [-dft_configuration_mode dft_config_mode]
 [-include_opcg_domain_information]
 [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.

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.

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:

rc:> 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>
  latch 5  DFT_Lockup_g1
  bit 6  out2_reg_4 <test_clk2/fall>
  ...
  bit 10 out2_reg_8 <test_clk2/fall>
  latch 10  DFT_Lockup_g348
  bit 11 out3_reg_4 <test_clk3/fall>
  ...

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

Add 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

Add 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.
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>
...      out2_reg_3 <test_clk2/rise>
llatch 23 DFT_Lockup_g350
bit 24    out3_reg_0 <test_clk3/rise>
...      out3_reg_3 <test_clk3/rise>

------------------------
rc:/> write_dft_abstract_model
scan style is muxed_scan

# writing abstract model for 1 scan chain

define_dft abstract_segment -module test \
-module 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

See the following sections in Design for Test in Encounter RTL Compiler:

- Creating a Scan Abstract Model
- Block-Level Domain Blocking Flow
- Hierarchical Compression Flow
- Bottom-Up Test Synthesis Flow

Affected by these commands:  
compress_block_level_chains on page 657  
connect_scan_chains on page 681

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 875  
write_hdl on page 276 (-abstract)
write_dft_rtl_model

write_dft_rtl_model
    -directory directory
    [design]

Writes out an RTL model of the design in Verilog if the DFT RTL insertion update flow is enabled. This command minimally modifies the user-supplied RTL files to include the RTL constructs of the inserted JTAG macro and MBIST structures.

Options and Arguments

design Specifies the design for which to update the RTL files.
    If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

-directory directory
    Specifies the directory to which the updated RTL files must be written. The directory is created if it does not exist.

Related Information

MBIST Top-Down RTL Insertion Flow in Design for Test in Encounter RTL Compiler

Affected by these commands
insert_dft_jtag_macro on page 808
insert_dft_mbist on page 819

Affected by this attribute: dft_rtl_insertion
write_et_atpg

write_et_atpg

[-ncsim_library string] [-library string]
[-directory string] [-run_from_et_workdir]
[ -configuration_mode_order dft_configuration_mode_list]
[ -delay ]
[-opcg_mode opcg_mode]
[-library string]
[-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] [-hier_test_core][design]

Writes out the necessary files and the template run scripts to run Automatic Test Pattern Generator (ATPG) using the Encounter Test software.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

The generated scripts depend on the setting of the dft_true_time_flow root attribute.

- With the default setting of the attribute, the command generates run scripts for static ATPG and non-SDF based OPCG delay test flows.

- When you enable the true time flow by setting this attribute to true, the command will also generate the tt_setup file. The Encounter Test true_time command is used to generate the ATPG script. This option provides scripts for the following ATPG test flows:

  static ATPG
  OPCG Delay Test
  non-OPCG Delay Test
  SDF/SDC based Delay Test (OPCG or non-OPCG)
  RAM Sequential Delay Test
  Path Delay Test
  Iddq Test

  For the last four flows, you will need to make minor modifications to the tt_setup file written by write_et_atpg, and you will need to regenerate the scripts using the Encounter Test true_time command. For more information on the true_time script and tt_setup file, refer to the Encounter Test Automatic Test Pattern Generation User Guide.

Note: In the true timing flow, several options are not applicable, while the behavior of other options differs. See the option descriptions for more information.
By default, this command generates the following files:

- **et.exclude**—A file listing objects to be excluded from the ATPG analysis in an assumed scan mode.

  **Note:** Because the true time flow does not support the assumed scan mode, this file is not written out in the true time flow.

- **et.modedef**—A mode definition file, a text file that describes the test mode in assumed scan mode

- **topmodulename Assumes 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.

  **Note:** Because the true time flow does not support the assumed scan mode, this file is not written out in the true time flow.

- **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.

- **runet.atpg**—A template script file to run the requested testability analysis

- **topmodulename.et_netlist.v**—A netlist for Encounter Test

If the **write_et_atpg** command is run with the -compression option, 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 when inserting broadcast-based decompression logic

If the **write_et_atpg** command is specified with the -delay option, the following files are generated in addition to the **topmodulename FULLSCAN pinassign** file:

- **topmodulename FULLSCAN OPCGModeName pinassign**—A pin assignment file generated only when inserting OPCG logic in full scan mode

  This file includes the OPCG test signals with their appropriate test functions and includes the oscillator and domain related information.

- **topmodulename FULLSCAN OPCGModeName seqdef**—A sequence definition file generated when inserting OPCG logic in full scan mode.
This file is used to initialize the PLL and establish the OPCG mode.

All of the files are used to validate their specific test mode.

- If the `write_et_atpg` command is specified with the `-compression` and `-delay` options, the following files are generated in addition to the `topmodulename.FULLSCAN.pinassign` and `topmodulename.COMPRESSION.pinassign` files.

  - `topmodulename.COMPRESSION_OPCGModeName.pinassign`—A pin assignment file generated only when inserting OPCG logic with XOR-based decompression logic. This file includes the OPCG test signals with their appropriate test functions and includes the oscillator and domain related information.

  - `topmodulename.COMPRESSION_OPCGModeName.seqdef`—A sequence definition file generated when inserting OPCG logic with XOR-based compression logic. This file is used to initialize the PLL and establish the OPCG mode.

All of the files are used to validate their specific test mode.

- If the `write_et_atpg` command is specified with the `-hier_test_core` option, files are written to generate migratable patterns for a Core. The design must have been taken through the “Preparing a Core” flow.

  - `run_compression_decomp_sim`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test for compression logic built using XOR-based decompression logic.

  - `run_compression_sim`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test for compression logic built using broadcast-based decompression logic.

  - `run_fullscan_sim`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test in full scan mode.

  - `run_fullscan_sim_OPCGModeName`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test in full scan mode with OPCG logic.

  - `run_compression_sim_OPCGModeName`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test for compression logic built using XOR-based decompression logic and with OPCG logic.

Note: Some files can be customized according to the setup requirements.
In the true flow, the command writes out all files related to all configuration modes in the design.

- If the `write_et_atpg` command is specified with the `-opcg_mode` option, the following files are written out in addition:
  - `topModuleName.FULLSCAN_OPCGModename.pinassign`
  - `topModuleName.FULLSCAN_OPCGModename.seqdef`

- If you also specified the `-compression` option, the following files are also written:
  - `topmodulename.COMPRESSION_OPCGModeName.pinassign`
  - `topmodulename.COMPRESSION_OPCGModeName.seqdef`

**Options and Arguments**

- **-atpg_options {option1=value option2=value}**
  
  Specifies extra ATPG analysis options.
  
  **Note:** This option does not apply to the true time flow.

- **-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 Encounter Test documentation).

  **Note:** This option does not apply to the true time flow.

- **-build_model_options {option1=value option2=value}**

  Specifies extra options to apply when building the Encounter Test model.

  **Note:** For more information on these options, refer to the `build_model` command in the *Command Line Reference* (of the Encounter Test 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 for Encounter Test.
**Note:** For more information on these options, refer to the `build_testmode` command in the *Command Line Reference* (of the Encounter Test documentation).

**Note:** This option does not apply to the true time flow.

**-compression**

Instructs to write out the files needed to run ATPG-based testability analysis for compression mode.

**-configuration_mode_order** `dft_configuration_mode...`

Specifies to write Encounter Test script files for a compression mode. Valid compression mode names are: COMPRESSION, COMPRESSION_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.

**-continue_with_severe**

Allows the continuation of the Encounter Test script even when severe warnings occur during execution of the commands in the script.

**-delay**

Specifies to generate additional files for Encounter Test to verify the OPCG logic.

- In the default flow, you cannot combine this option with the `configuration_mode_order` option.
- In the true time flow, you can combine this option with the `configuration_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 Encounter Test input files.

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(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_working_directory/et_scripts

-force

Specifies to continue even if the JTAG macro used to enable the JTAG-controlled compression macro is not found.

-hier_test_core

Instructs to write out the files needed to generate migratable patterns for a core.

-library string

Specifies the list of Verilog structural library files required to run Encounter Test ATPG.

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:

```plaintext
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_et_atpg` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_et_atpg -library "./padcells.v ./stdcells.v ./memories ./ip_blocks" ...
```

2. If using an include file. Create an include file named include_libraries.v containing:

```plaintext
'include "./padcells.v"
'include "./stdcells.v"
```

And then specify the following:

```
write_et_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 Encounter Test will be run.
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:

  ```
  rc:/> 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
  ```

-ncsim_library string

  Specifies the list of library files required for the Incisive Enterprise simulation of the generated vectors.

  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_et_workdir

  Allows to run the Automatic Test Pattern Generator (ATPG) from the working directory specified with the `-directory` option.

  **Default:** parent directory of the specified work directory.

-verify_test_structures_options {option1=value option2=value}

  Specifies extra options to apply when performing test structure verification for Encounter Test.

  **Note:** This option does not apply to the true time flow.
Related Information

See the following sections in Design for Test in Encounter RTL Compiler:

- Using Encounter Test 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 Encounter Test in “Inserting Scan Chain Compression Logic”
- Using Encounter Test to Perform a Deterministic Fault Analysis on a Scan Connected Netlist
- Hierarchical Test Flow: Preparing a Core in “Hierarchical Test”
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_et_workdir]
   [-continue_with_severe] [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.et_netlist.v—A netlist for Encounter Test
■ runet.bsv—An Encounter Test run file to run Boundary Scan Verification (BSV).

Note: Some files can be customized according to the setup requirements.

For more information on the exact Encounter Test product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.
-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 Encounter Test model.

-continue_with_severe
   Allows the continuation of the Encounter 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 Encounter Test 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/et_scripts
-library string

Specifies the list of Verilog structural library files.

The Verilog libraries required to run Encounter Test ATPG and Incisive Enterprise 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.

For example, if the Verilog files required to run ATPG and Incisive Enterprise 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 Encounter Test will be run.

-run_from_et_workdir

Allows to run boundary scan verification from the working directory specified with the -directory option.

Default: parent directory of the specified work directory.
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:
  
  ```
  rc:/> shell ls bsv
  topmodulename.bsdl
  runet.bsv
  test.et_netlist.v
  test.rc_netlist.v
  ```

Related Information

[Generating Script for Boundary Scan Verification in Design for Test in Encounter RTL Compiler](Design for Test in Encounter RTL Compiler)
write_et_dfa

write_et_dfa
    [-library string]
    [-build_model_options string]
    [-build_testmode_options string]
    [-atpg_options string] [-dfa_options string]
    [-include_redundant_faults]
    [-verify_test_structures_options string]
    [-directory string] [-run_from_et_workdir]
    [-min_slack_for_no_tp_file integer]
    [-continue_with_severe] [design]

Writes out the necessary files and the template run scripts to run Deterministic Fault Analysis using the Encounter Test 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
- `runet.dfa`—A template script file to run the requested deterministic fault analysis
- `topmodulename.et_netlist.v`—A netlist for Encounter Test
- `TestPointInsertion.testmode_name.dfa`—A file containing test point locations.
- `TestPointInsertion.FULLSCAN_inactive.dfa`—A file containing the additional test point locations found when the inactive faults are included during DFA analysis
- `run_fullscan_sim`—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test in full scan mode.

Note: Some files can be customized according to the setup requirements.

For more information on the exact Encounter Test product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.
Options and Arguments

-atpg_options {option1=value option2=value}
   Specifies extra ATPG analysis options.

-build_model_options {option1=value option2=value}
   Specifies extra options to apply when building the Encounter Test model.

-build_testmode_options {option1=value option2=value}
   Specifies extra options to apply when building the test mode for Encounter Test.

-continue_with_severe
   Allows the continuation of the Encounter 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 Encounter Test 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/et_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 Encounter Test ATPG and Incisive Enterprise simulation of the generated vectors must be provided using the -library option of the write_et_dfa command. These libraries must be in a structural format. The files can be specified separately on the write_et_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.
For example, if the Verilog files required to run ATPG and Incisive Enterprise simulation are:

```
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_et_dfa` can be used in either of the following ways:

1. If specifying files separately on the command line:

```
write_et_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_et_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 Encounter Test 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_et_no_tp_file` root attribute is set to `true`, the tool adds the pin to a file that prevents Encounter Test from inserting a test point on such pins.

```
-run_from_et_workdir
```

Allows to run the Deterministic Fault Analysis from the working directory specified with the `-directory` option.

**Default:** parent directory of the specified work directory.

```
-verify_test_structures_options {option1=value option2=value}
```

Specifies extra options to apply when performing test structure verification for Encounter Test.
Examples

- The following command generates the files to run deterministic fault analysis.

```bash
write_et_dfa -directory dfa -library $sim/mylib.v
```

Examining the `dfa` directory that was generated shows the following files:

```bash
rc:/> shell ls dfa
runet.dfa
run_fullscan_sim
test.et_netlist.v
test.FULLSCAN.pinassign
test.rc_netlist.v
```

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Exporting the Design for Test Point Selection
- Using Encounter Test to Perform a Deterministic Fault Analysis on a Scan Connected Netlist

Related command: `insert_dft dfa_test_points` on page 804

Affected by this attribute: `dft_generate_et_no_testpoint_file`
write_et_lbist

write_et_lbist
   [-library string] [-directory string]
   [-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}]
   [-run_from_et_workdir] [design]

Writes out data and script files for Encounter Test 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 Encounter Test
- *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—An Encounter Test file to run LBIST tests for RUNBIST/SETBIST instruction
- *run_sim_*.instructionName—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test for LBIST.

This command generates the following files for an LBIST macro using direct access interface:

- *topmoduleName*.et_netlist.v—A netlist for Encounter Test
- *assignfile.MODE_LBIST_DIRECT_*—A pin-assignment file for the the LBIST testmode
assignfile.NCSIM—A pin assignment file for the testmode used to generate verilog testbench 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—An Encounter Test file to run direct-access LBIST tests

run_sim_DIRECT—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test for for direct-access LBIST.

run_ET_NCSIM—An Encounter Test file to generate the Verilog testbench to verify the LBIST signature

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

-buildFaultmodel_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 Encounter Test documentation).

-build_model_options {option1=value option2=value ...}

Specifies extra options to apply when building the Encounter Test model.

Note: For more information on these options, refer to the build_model command in the Command Line Reference (of the Encounter Test documentation).

-build_testmode_options {option1=value option2=value ...}

Specifies extra options to apply when building the test mode for Encounter Test.
Note: For more information on these options, refer to the build_testmode command in the Command Line Reference (of the Encounter Test documentation).

design

Specifies the design for which to write out the Encounter Test input files.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

-dictionary string

Specifies the directory to which the output files must be written.

Default: current_working_directory/et_scripts

-library string

Specifications the list of Verilog structural library files required to run Encounter Test.

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 Encounter Test will be run.

-run_from_et_workdir

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.

-verify_test_structures_options {option1=value option2=value}

Specifies extra options to apply when performing test structure verification for Encounter Test.

Related Information

See the following sections in Design for Test in Encounter RTL Compiler:

- Inserting LBIST Logic
- Generating Files for LBIST Pattern Generation and Simulation

Affected by these commands: insert_dft_logic_bist on page 813
                        write_logic_bist_macro on page 954
write_et_mbist

write_et_mbist
- mbist_interface_file_dir string
- mbist_interface_file_list string
[-build_model_options string]
[-create_embedded_test_options string]
[-bsv [-bsdl file [-bsdl_package_path string]
     [-bsdl_package_name files] ] [-library string]
[-directory string] [-run_from_et_workdir]
[-force] [-continue_with_severe] [design]

Writes out the necessary files and the template run scripts to run Create Embedded Test using the Encounter Test software.

This command generates the following files:

- 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.bsdl—A BSDL file produced when specifying the -bsv and BSDL-related options.

- runet.mbist—An Encounter Test run file to run Boundary Scan Verification (BSV) when specifying the -bsv option.

- topmodulename.et_netlist.v—A netlist for Encounter Test

- runet.mbist_interface—A template script file to run Create Embedded Test in Encounter Test

Note: Some files can be customized according to the setup requirements.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

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.

- **-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.

- **-bsv**

  Writes out the files needed for boundary scan verification.

  **Note:** This option prints a pin assignment file if differential PAD pairs are detected in the design.

- **-build_model_options** {*option1=value option2=value*}

  Specifies extra options to apply when building the Encounter Test model.

- **-continue_with_severe**

  Allows the continuation of the Encounter Test script even when severe warnings occur during execution of the commands in the script.
-create_embedded_test_options {option1=value option2=value}

Specifies extra options to apply when running Create Embedded Test in Encounter Test.

design

Specifies the design for which to write out the Encounter Test 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: currentWorkingDirectory/et_scripts

-force

Writes out the scripts when MBIST was inserted using direct access mode.

Normally this command requires a JTAG macro to generate the scripts. If you insert MBIST using the direct access mode, you must use the -force option to successfully complete the command.

-library string

Specifies the list of Verilog structural library files.

The Verilog libraries required to run Encounter Test ATPG and Incisive Enterprise simulation of the generated vectors must be provided using the -library option of the write_et_mbist command. These libraries must be in a structural format. The files can be specified separately on the write_et_mbist 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.

For example, if the Verilog files required to run ATPG and Incisive Enterprise simulation are:

./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v

write_et_mbist can be used in either of the following ways:

1. If specifying files separately on the command line:

   write_et_mbist -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_mbist -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 Encounter Test will be run.

- **-mbist_interface_file_dir**
  
  Specifies the MBIST interface file directories. Separate the directory names with blank spaces.

- **-mbist_interface_file_list**
  
  Specifies a list of MBIST interface files. Separate the file names with commas, for example, `file1,file2`.

- **-run_from_et_workdir**
  
  Allows to run Create Embedded Test from the working directory specified with the `-directory` option.
  
  **Default:** parent directory of the specified work directory.

**Examples**

- The following command generates the files to run Boundary Scan Verification and Create Embedded Test in Encounter Test.

```
write_et_mbist -mbist_interface_file_dir directory \ 
   -mbist_interface_file_list file1,file2 -bsv -library $sim/mylib.v
```

Examining the `atpg` directory that was generated shows the following files:

```
rc:/> shell ls mbist
test.COMPRESSION_DECOMP.pinassign
test.COMPRESSION.pinassign
test.et_netlist.v
test.FULLSCAN.pinassign
test.rc_netlist.v
runet.mbist
runet.mbist_interface
```
Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*:

- Writing the Scripts and Setup Files to Generate MBIST Patterns
- MBIST Top-Down Gate Insertion Flow
write_et_no_tp_file

write_et_no_tp_file

[\text{-min\_slack integer}] \text{[design]}

Generates a \text{design.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 Encounter Test via the \text{notpfile} keyword of the \text{analyze_random_resistance} and \text{analyze_deterministic_faults} commands. With this information, Encounter Test 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 \text{dft\_dont\_scan} attribute set to \text{true}
- The subdesign or instance has the \text{preserve} attribute set to \text{true, 1, or size\_ok}
- The instance has the \text{hard\_region} attribute set to \text{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 \text{min\_slack}
- The pin has the \text{preserve} attribute set to \text{true, 1, or size\_ok}

In addition, any net that does not have its \text{preserve} attribute set to \text{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

\text{design} \quad \text{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.

\text{-min\_slack integer} \quad \text{Prevents insertion of a test point if the slack is smaller than the specified number (in ps).}
write_et_rrfa

write_et_rrfa
  [-atpg] [-force] [-atpg_options string]
  [-rrfa_options string] [-build_model_options string]
  [-build_testmode_options string]
  [-verify_test_structures_options string]
  [-library string]
  [-directory string] [-run_from_et_workdir]
  [-min_slack_for_no_tp_file integer]
  [-continue_with_severe] [design]

Writes out the necessary files and the template run scripts to run either Automatic Test Pattern Generator (ATPG) or Random Resistance Fault Analysis (RRFA) based testability analysis, generate test patterns using the Encounter Test software.

This command generates the following files:

- **et.exclude**—A file listing objects to be excluded from the ATPG analysis in an assumed scan mode.
- **et.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
- **run_fullscan_sim**—An Incisive Enterprise simulator run file used to simulate the test patterns created by Encounter Test in full scan mode.

**Note:** Some files can be customized according to the setup requirements.

For more information on the exact Encounter Test product requirements, refer to [Encounter Test Product Requirements for Advanced Features](#) in Design for Test in Encounter RTL Compiler.
Options and Arguments

- **atpg**
  Writes out the files needed to run Automatic Test Pattern Generation using the Encounter Test software.

- **atpg_options** (option1=value option2=value)
  Specifies extra ATPG analysis options.

- **build_model_options** (option1=value option2=value)
  Specifies extra options to apply when building the Encounter Test model.

- **build_testmode_options** (option1=value option2=value)
  Specifies extra options to apply when building the test mode for Encounter Test.

- **continue_with_severe**
  Allows the continuation of the Encounter 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 Encounter Test 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/et_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 Encounter Test ATPG and Incisive Enterprise simulation of the generated vectors must be provided using the -library option of the write_et_rrfa command. These libraries must be in a structural format. The files can be specified separately on the write_et_rrfa 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.
For example, if the Verilog files required to run ATPG and Incisive Enterprise simulation are:

```plaintext
./padcells.v
./stdcells.v
./memories/*.v
./ip_blocks/*.v
```

`write_et_rrfa` can be used in either of the following ways:

1. If specifying files separately on the command line:

   ```plaintext
   write_et_rrfa -library "./padcells.v ./stdcells.v \n./memories ./ip_blocks" ...
   ```

2. If using an include file. Create an include file named `include_libraries.v` containing:

   ```plaintext
   'include "./padcells.v"
   'include "./stdcells.v"
   ```

   And then specify the following:

   ```plaintext
   write_et_rrfa -library "include_libraries.v \n./memories ./ip_blocks" ...
   ```

   **Note:** If you specify a relative path, the command interprets the path to be the location from where Encounter Test 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_et_no_tp_file` root attribute is set to `true`, the tool adds the pin to a file that prevents Encounter Test from inserting a test point on such pins.

   `-rrfa_options string`

   Specifies the extra options to run RRFA-based testability analysis in a string.

   `-run_from_et_workdir`

   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.
-verify_test_structures_options {option1=value option2=value}

Specifies extra options to apply when performing test structure verification for Encounter Test.

Examples

- The following command generates the files to run an ATPG-based testability analysis.

  `write_et_rrfa -atpg -directory atpg -library $sim/mylib.v`

  Examining the `atpg` directory that was generated shows the following files:

  ```shell
  shell> ls rrfa
  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
  ```

Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*

- Using Encounter Test to Automatically Select and Insert Test Points
- Exporting the Design for Test Point Selection

Affected by this attribute `dft_generate_et_no_testpoint_file`
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 RTL Compiler 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 RTL Compiler session.

Note: The recommended approach to completely restoring the DFT setup (including the boundary scan objects) in a new RTL Compiler session is to use the write_script/read_netlist approach.
Related Information

Writing the IOSPecList in Design for Test in Encounter RTL Compiler

Affected by these commands:

- `define_dft_jtag_instruction` on page 708
- `define_dft_jtag_instruction_register` on page 712
- `insert_dft_boundary_scan` on page 788
- `insert_dft_mbist` on page 819
- `insert_dft_ptam` on page 833

Related command:

- `read_io_speclist` on page 878
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_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]
  [- programmable_defaults] [- 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 Encounter Test license. For more information on the exact product requirements, refer to [Encounter Test Product Requirements for Advanced Features](#) in Design for Test in Encounter RTL Compiler.

For more information about LBIST, contact your local Cadence representative.

**Options and Arguments**

- **-add_direct_access_support**
  
  Inserts the LBIST macro with direct-access support.

- **-add_masking**
  
  Inserts the LBIST macro including the optional channel masking logic (only available with SETBIST support).

  **Note:** This option can only be specified with the -add_setbist_support option.

- **-add_runbist_support**
  
  Inserts the LBIST macro with JTAG RUNBIST support.
-add_setbist_support
  Inserts the LBIST macro with JTAG SETBIST support.

-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.

-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 Encounter Test.

-max_length_of_channels integer
  Specifies the length of the longest compressed scan channel.
-programmable_defaults
Generates a parallel interface on the LBIST macro that can be used to configure LBIST parameters externally, even at run-time. Using this option, the RUNBIST and direct-access interfaces can achieve a level of control similar to the SETBIST interface, but without requiring a serial load of the parameters.

The generated interface allows setting all available counter and window values as well as PRPG and mask values.

When this option is omitted, these parameters are assumed to be constant and are hard-coded inside the macro, saving area but reducing flexibility.

-reset_patterns integer
Specifies the default number of reset test patterns to be executed.

-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_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
  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 Design for Test in Encounter RTL Compiler

Related commands: insert_dft_logic_bist on page 813
write_et_lbist on page 938
write_mbist_testbench

write_mbist_testbench
[-create_embedded_test_options string]
[-irun_options string] [-ncsim_library file_list]
[-testbench_directory string] [-directory string]
[-no_deposit_script] [-script_only] [-no_build_model]
[design]

Writes out the necessary files and the template run scripts to create Verilog test benches to validate memory BIST, and executes the scripts using the Encounter Test software.

If your design contains ROMs, include the rompath and romcontentsfile keywords in the create_embedded_test_options string.

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 MBIST into the design and optionally a JTAG macro, write out the modified design to file. Then, use this command to generate an MBIST Verilog test bench for either a gate-level or RTL netlist to validate the MBIST functionality through simulation.

This command is designed to generate the following patterns by default:

- Bypass and production patterns— if JTAG control has been implemented for memory BIST in the design
- Poweron and burnin patterns—if direct access has been implemented for memory BIST 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 Encounter Test:

- design_abstract.v—A Verilog generic logic gates description of the netlist for Encounter Test
- runet.write_mbist_testbench—A template script file to run Create Embedded Test in Encounter Test, generating patterns for memory BIST
- runet.write_vectors—A template script file to run Write Vectors in Encounter Test to write the memory BIST patterns as Verilog test benches
irun.simscript.testbench_pattern_name—One or more template scripts to compile and simulate the MBIST test bench pattern within the Incisive Enterprise simulator. By default, these scripts will be for BYPASS and Production patterns when JTAG is used, and for Poweron and Burnin patterns when direct access is used.

You can customize some files according to the setup requirements.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to [Encounter Test Product Requirements for Advanced Features](#) in *Design for Test in Encounter RTL Compiler*.

### Options and Arguments

- **-create_embedded_test_options** `{option1=value option2=value}`
  
  Specifies extra options to apply when running Create Embedded Test in Encounter Test.

- **design**
  
  Specifies the design for which to write out the Encounter Test 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/wmt`

- **-irun_options** `{option1=value option2=value ...}`
  
  Specifies the options to be passed to the `irun` command for simulation purposes.

- **-ncsim_library file_list**
  
  Specifies the list of library files required for the Incisive Enterprise 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_mbist_testbench can be used in either of the following ways:

1. If specifying files separately on the command line:

   write_mbist_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:

   `include "./padcells.v"
   `include "./stdcells.v"

   And then specify the following:

   write_mbist_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 Incisive Enterprise simulator will be run.

-script_only

Creates the runet.write_mbist_testbench 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.

-no_build_model

Excludes the build_model command from the script generated to execute the Encounter Test software.

If you use the -create_embedded_test_options option with the jtagtestmode or pmdatestmode keyword, the generated script will automatically exclude the build_model command.

-no_deposit_script

Prevents that the deposit script is generated and used during simulation.

-testbench_directory directory

Specifies the directory to which the generated Verilog test benches must be written.

Default: current_working_directory/mbist_testbench
Examples

- In the following example a single MBIST instruction set was implemented in the design:

  ```
  write_mbist_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 (MBIST was inserted on multiple designs or blocks) using separate JTAG instructions to access each block's MBIST engines.

  ```
  write_mbist_testbench \
  -create_embedded_test_options \ 
  interfacefilelist="BLOCK_pattern_control.txt, \ 
  BLOCK_mbist_tdr_map.txt,BLOCK_mbistdiag_tdr_map.txt" \ 
  -testBench_directory ./mbist_verilog_testbenches \ 
  -ncsim_library../$simulation_verilog_libraries \ 
  -directory ./mbist_verification_scripts
  ```

- The following example shows how to specify the command when ROMS are present in the design:

  ```
  write_mbist_testbench \
  -Create_embedded_test_options \ 
  "rompath=./memory_data romcontentsfile=ROM256x34.hex " \ 
  -testbench_directory ./mbist_verilog_testbenches \ 
  -ncsim_library../$simulation_verilog_libraries \ 
  -directory ./mbist_verification_scripts
  ```

Related Information

Design Flows in “Inserting Memory Built-In-Self-Test Logic” in Design for Test in Encounter RTL Compiler

Affected by these commands

- `define_dft mbist_direct_access` on page 722
- `insert_dft boundary_scan` on page 788
- `insert_dft jtag_macro` on page 808
- `insert_dft mbist` on page 819

Related commands:

- `write_dft_rtl_model` on page 921
- `write_et_bsv` on page 930
- `write_et_mbist` on page 942
- `write_hdl` on page 276

Related attributes:

- `dft_rtl_insertion`
- `mbist_instruction_set`
write_pmbist_interface_files

write_pmbist_interface_files
  -directory string [design]

Writes out the necessary interface files for programmable Memory Built-In-Self-Test (PMBIST) related to this design.

RTL Compiler 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 Encounter Test 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 are required.
- 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.

**Note:** To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.
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.

Related Information

Affects this command: write_pmbist_interface_files on page 962

Related command: read_pmbist_interface_files on page 881
write_pmbist_testbench

write_pmbist_testbench
[-create_embedded_test_options string] [-irun_options string] [-ncsim_library file_list]
[-testbench_directory string] [-directory string]
[-no_deposit_script] [-script_only] [no_build_model] [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 Encounter Test 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 Encounter Test:

- design_abstract.v—A Verilog generic logic gates description of the netlist for Encounter Test
- runet.write_mbist_testbench—A template script file to run Create Embedded Test in Encounter Test, generating patterns for programmable memory BIST
- runet.write_vectors—A template script file to run Write Vectors in Encounter Test to write the programmable memory BIST patterns as Verilog test benches
irun.simscript.testbench_pattern_name—One or more template scripts to compile and simulate the PMBIST test bench pattern within the Incisive Enterprise 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.

irun.depositscript.testbench_pattern_name—One or more Incisive Enterprise simulator scripts to initialize all PMBIST and JTAG logic to known states prior to executing the Verilog test benches. RTL Compiler 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.

You can customize some files according to the setup requirements.

Note: To use this command you need an Encounter Test license. For more information on the exact product requirements, refer to Encounter Test Product Requirements for Advanced Features in Design for Test in Encounter RTL Compiler.

Options and Arguments

- -create_embedded_test_options {option1=value option2=value}
  Specifies extra options to apply when running Create Embedded Test in Encounter Test.

design
  Specifies the design for which to write out the Encounter Test 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/wmt

- -irun_options {option1=value option2=value ...}
  Specifies the options to be passed to the irun command for simulation purposes.

- -ncsim_library file_list
  Specifies the list of library files required for the Incisive Enterprise 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:
   
   ```
   '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 Incisive Enterprise simulator will be run.

   **-no_build_model** Excludes the build_model command from the script generated to execute the Encounter Test software.

   If you use the -create_embedded_test_options option with the jtagtestmode or pmdatestmode keyword, the generated script will automatically exclude the build_model command.

   **-no_deposit_script** Prevents that the deposit script is generated and used during simulation.

   **-script_only** Creates the runet.write_mbist_testbench 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.

   **-testbench_directory directory** Specifies the directory to which the generated Verilog test benches must be written.
Examples

In the following example a single PMBIST instruction set was implemented in the design:

```bash
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.

```bash
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 734  
- `insert_dft boundary_scan` on page 788  
- `insert_dft jtag_macro` on page 808  
- `insert_dft pmbist` on page 827

Related commands:  
- `read_hdl` on page 216  
- `write_pmbist_interface_files` on page 962

Related attribute:  
- `pmbist_instruction_set`
**write_scandef**

`write_scandef [-partition partition -chains chain [chain]...]... [-version {5.4|5.5}] [-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.
-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 {5.4|5.5}

 Specifies which DEF version to write out. Version 5.5 writes out the MAXBITS and PARTITION keywords as regular statements (that is, uncommented).

 Default: 5.4
Example

The following example writes out the scanDEF information to the screen:

```plaintext
rc:/> write_scandef

VERSION 5.4 ;
NAMECASESENSITIVE 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

Creating a scanDEF File in Design for Test in Encounter RTL Compiler

Affected by this command: connect_scan_chains on page 681

Affected by these attributes: Actual Scan Chain attributes
Low Power Synthesis

- `build_rtl_power_models` on page 973
- `clock_gating` on page 975
- `clock_gating_connect_test` on page 977
- `clock_gating_declone` on page 978
- `clock_gating_import` on page 979
- `clock_gating_insert_in_netlist` on page 981
- `clock_gating_insert_obs` on page 982
- `clock_gating_join` on page 984
- `clock_gating_remove` on page 986
- `clock_gating_share` on page 988
- `clock_gating_split` on page 990
- `read_saif` on page 992
- `read_tcf` on page 997
- `read_vcd` on page 1002
- `report_clock_gating` on page 1006
- `report_power` on page 1007
- `state_retention` on page 1008
- `state_retention_connect_power_gating_pins` on page 1009
- `state_retention_swap` on page 1010
- `write_forward_saif` on page 1011
- `write_saif` on page 1013
write_tcf on page 1015
build_rtl_power_models

build_rtl_power_models
  [-clean_up_netlist]
  [-relative instance_list]
  [-design design]

Builds detailed power models for more accurate RTL power analysis. The models are used in subsequent RTL power analysis reports.

**Note:** You can only run this command after you have executed either the `synthesize -to_generic` or `synthesize -to_clock_gated` command.

The power models are MSV and PSO aware.

If you have super-threading enabled, it will be used for power model building.

**Options and Arguments**

- **-clean_up_netlist** Requests to remove unreachable logic in the netlist as this can affect the accuracy of the estimation.
  
  **Note:** Unreachable logic is removed from the input netlist, without changing the logic functionality.

- **-design design** Specifies the design for which to build the power models.

- **-relative instance_list** Builds separate power models for each of the specified hierarchical instances. Models for the top design are built separately at the end.

**Examples**

- The following example shows an extract of the messages that are printed in the log file when you build the RTL power models.

```
rc:/> build_rtl_power_models -clean_up_netlist

... Cleaning up the design /designs/mult_bit_muxed_add ...
Starting building RTL power analysis models ...
Preprocessing the netlist for building RTL power models ...
Building RTL power models for top-level design /designs/mult_bit_muxed_add ...
Done building models for power analysis.
RTL power modeling has finished. Use command 'report power' to see power report.
```
The following example shows the messages that are printed in the log file when you build separate RTL power models for hierarchical instances.

```
rc:/> build_rtl_power_models -clean_up_netlist -relative {mult_1 mult_2}
Cleaning up the design /designs/test ...
Starting building RTL power analysis models ...
Preprocessing the netlist for building RTL power models ...
Building RTL power models for domain /designs/test/instances_hier/mult_1 ...
Building RTL power models for domain /designs/test/instances_hier/mult_2 ...
Building RTL power models for top-level design /designs/test ...
Done building models for power analysis.
RTL power modeling has finished. Use command ’report power’ to see power report.
```

Related Information

**RTL Power Analysis** in *Low Power in Encounter RTL Compiler*

Related command:  
report power on page 1007

Affected by this attribute:  
lp_insert_clock_gating
**clock_gating**

```plaintext
clock_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**

- `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 RC-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.
- `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.

**Related Information**

Related commands:  
- `clock_gating connect_test` on page 977  
- `clock_gating declone` on page 978  
- `clock_gating import` on page 979
Command Reference for Encounter RTL Compiler
Low Power Synthesis

clock_gating insert_in_netlist on page 981

clock_gating insert_obs on page 982

clock_gating join on page 984

clock_gating remove on page 986

clock_gating share on page 988

clock_gating split on page 990
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 RC-LP engine.

**Example**

- The following command connects the test inputs connected to Test1.

  ```
synthesize
  ...
  set_att lp_clock_gating_test_signal Test1
  ...
  clock_gating connect_test
  ```

**Related Information**

- Clock Gating with DFT in *Low Power in Encounter RTL Compiler*
- Scan Insertion after Clock-Gating Insertion in *Low Power in Encounter RTL Compiler*

  Related command: `report clock_gating` on page 1006

  Affected by the attribute: `lp_clock_gating_test_signal`
clock_gating declone

clock_gating declone
     [-hierarchical] [-no_clock_tree_traversal]
     [-verbose]

Merges clock-gating instances driven by the same inputs. The RC-LP engine automatically
removes any dangling ports.

This command is register-bank aware and power-domain aware.

Note: This command works only on a netlist whose clock-gating logic was inserted by the
RC-LP engine.

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.

If you do not set this option, RTL Compiler 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

Decloning Clock-Gating Instances in Low Power in Encounter RTL Compiler

Related command: report clock_gating on page 1006

Affected by this attribute: lp_clock_gating_max_flops
clock_gating 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 RC-LP engine.

The command returns the total number of instances imported.

Currently, the RC-LP engine recognizes the following structures as clock-gating instances:

- 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 RC-LP engine does not recognize the gate as a clock-gating instance.

- Integrated clock-gating cells

Currently, the RC-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 RC-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 RC-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 RTL Compiler.

  By default, this command only affects instances at the current level of the design hierarchy.

- **-start_from instance**
  Starts processing clock-gating instances that were not inserted by RTL Compiler 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.

  rc:/> 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.

  rc:/> 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 1006
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**

See the following sections in Low Power in Encounter RTL Compiler

- Clock Gating and Scan Chain Insertion in Mapped Netlist
- Clock-Gating Insertion in a Scan-Connected Netlist

**Related command:** report clock_gating on page 1006

**Affected by this attribute:** lp_clock_gating_test_signal
**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 RC-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 RC-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.

- **-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**

See the following sections in Low Power in Encounter RTL Compiler

- **Clock Gating with DFT**
- **Clock Gating and Scan Chain Insertion in Mapped Netlist**
- **Recommended Bottom-Up Clock Gating Flow with DFT**
- **Scan Insertion after Clock-Gating Insertion**
- **Clock-Gating Insertion in a Scan-Connected Netlist**

Related commands: define_clock on page 318

report clock gating on page 1006

Affected by this attribute: lp_clock_gating_add_obs_port
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 RC-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
Related Information

Consolidating Multi-Stage Clock-Gating Logic in *Low Power in Encounter RTL Compiler*

Related command: `report clock gating` on page 1006
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 RC-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 RC-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 RC-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.
Example

- The following command removes all clock-gating instances in the hierarchy of subdesign sub1.
  
  ```
  rc:/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.
  
  ```
  rc:/> 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

Removing Clock-Gating Instances in *Low Power in Encounter RTL Compiler*

Related command: `report clock gating` on page 1006
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 RC-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.
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

**Creating Shared Clock Gating Logic Using Common Enable** in *Low Power in Encounter RTL Compiler*

Related command: `report clock_gating` on page 1006
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 RC-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 RC-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]
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
```
**read_saif**

*read_saif* [-scale *scale_factor*]
  [-update [-weight *weight_factor*]]
  [-verbose] [-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:

  \[
  \text{prob\_new} = \frac{\text{prob\_old} + w \times \text{prob\_spec}}{1+w} \\
  \text{tr\_new} = \frac{\text{tr\_old} + w \times \text{tc\_spec}/\text{duration\_spec}}{1+w}
  \]

  where \(\text{prob\_old}\) and \(\text{tr\_old}\) are the stored values, and \(\text{prob\_spec}, \text{tc\_spec},\) and \(\text{duration\_spec}\) 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.

*-instance *instance* Reads in the switching activities for the specified instance.

  The instance name can refer to an instance in the design loaded in RC, or can refer to an instance name in the SAIF file.

- If the instance name refers to an instance in the design loaded in RTL Compiler, the RC-LP engine asserts switching activities on that instance in the loaded design.
In this case, a complete design is loaded in RTL Compiler. 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 RC-LP engine asserts switching activities on the design
  loaded in RTL Compiler.

  The design loaded in RTL Compiler 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 to 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.

```plaintext
-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*

```plaintext
-update
```

Indicates that you are updating the probability values and toggle
counts.

```plaintext
-verbose
```

Prints a message for each net that is asserted.

*Default: Silent mode. Prints the percent completion messages.*

```plaintext
-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:

rc:/> 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
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:

```
rc:/> read_saif and2.saif -scale 2.0
```

Check the asserted toggle rates on nets `in1` and `in2`:

```
rc:/> get_attr lp_asserted_toggle_rate nets/in1
0.004500
rc:/> 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:

```plaintext
(SAFILE
 ... 
(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`:

```
rc:/> 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
```

Related Information

See the following sections in *Low Power in Encounter RTL Compiler*

- Reading Switching Activity Information from a SAIF File
- Checking System Messages when Reading Switching Activities

Affects this command: `report power` on page 1007

Related command: `write_saif` on page 1013

Sets these attributes: `lp_asserted_probability`
lp_asserted_toggle_rate

Related attributes:
lp_probability_type
lp_toggle_rate_type
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:

\[
\begin{align*}
\text{prob}\_\text{new} &= \frac{\text{prob}\_\text{old} + w \times \text{prob}\_\text{spec}}{1+w} \\
\text{tr}\_\text{new} &= \frac{\text{tr}\_\text{old} + w \times \text{tc}\_\text{spec}/\text{duration}\_\text{spec}}{1+w}
\end{align*}
\]

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.
- `-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 RTL Compiler hierarchy to which the parsed TCF hierarchy (specified through the `-tcf_instance` option) corresponds.

For example, if a *partial* design is loaded in RTL Compiler but you have a TCF file that contains switching activities for the full design, the top design in RTL Compiler will correspond to an instance in the TCF hierarchy.

You can also have a full design loaded in RTL Compiler, but only have a partial TCF. In that case you need to specify the name of the instance in the RTL Compiler hierarchy to which the TCF file applies.

By default, the TCF file applies to the top design in the RTL Compiler 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 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.

---

### -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`. 
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:

```
rc:/> 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 : "0.750000e+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 make the toggle rates on all pins the same as in the previous example, use the following command:

```
rc:/> read_tcf -scale 2.0 example2.tcf
```

Default: 1.0
Consider the following TCF file (example1.tcf):

```tcl
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:

```
rc:> read_tcf example1.tcf
```

Now consider the following TCF file (example2.tcf):

```tcl
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:

```
rc:> read_tcf -update -weight 0.5 example2.tcf
```

You would get the same result by:

a. Creating the following TCF file (example3.tcf)

```tcl
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";
    }
  }
}
```

b. Reading the example3.tcf file using the following command:

```
read_tcf example3.tcf
```
Related Information

See the following sections in *Low Power in Encounter RTL Compiler*

- Reading Switching Activity Information from a TCF File
- Checking System Messages when Reading Switching Activities

**TCF Syntax** in *Toggle Count Format Reference*.

**Affects this command:** report power on page 1007

**Sets these attributes:**
- lp_asserted_probability
- lp_asserted_toggle_rate

**Related attributes:**
- lp_probability_type
- lp_toggle_rate_type
**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 module]
   [-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 RC-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.

  **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 RTL Compiler hierarchy to which the parsed VCD hierarchy (specified through the -vcd_scope option) corresponds.

For example, if a partial design is loaded in RTL Compiler but you have a VCD file that contains switching activities for the full design, the top design in RTL Compiler will correspond to an instance in the VCD hierarchy.

You can also have a full design loaded in RTL Compiler, but only have a partial VCD. In that case you need to specify the name of the instance in the RTL Compiler hierarchy to which the VCD file applies.

By default, the VCD file applies to the top design in the RTL Compiler 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.
-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 RC-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 module

Starts parsing the VCD hierarchy from the specified module. You can specify to start parsing from the top or from a particular module 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.
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

See the following sections in _Low Power in Encounter RTL Compiler_

- **Reading Switching Activity Information from a VCD File**
- **Checking System Messages when Reading Switching Activities**

Affects this command: `report power` on page 1007

Related attributes:

- `lp_dynamic_analysis_scope`
- `lp_power_analysis_effort`
report clock_gating

Refer to report clock_gating in Chapter 9, “Analysis and Report.”
report power

Refer to report power in Chapter 9, “Analysis and Report.”
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

State-Retention Cell Replacement when Starting with Mapped Netlist in Low Power in Encounter RTL Compiler

Related commands:

state_retention connect_power_gating_pins on page 1009
state_retention swap on page 1010
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

State-Retention Cell Replacement when Starting with Mapped Netlist in Low Power in Encounter RTL Compiler

Related attributes: power_gating_pin_class power_gating_pin_phase
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

State-Retention Cell Replacement when Starting with Mapped Netlist in Low Power in Encounter RTL Compiler

Affected by these attributes: hdl_enable_proc_name
                           hdl_proc_name
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
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 1031  
- `read_saif` on page 992
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 RC-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

-bounded_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.
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 992

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
write_tcf

write_tcf [-duration simulation_period] [-computed]
 [-hierarchical] [-include_hier_ports] [-pin] [> file]

Writes a TCF file containing the user-assigned 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.

- **hierarchy**
  Writes out a TCF file in hierarchical format.
  
  By default, the RC-LP engine writes out a flat TCF file.

- **include_hier_ports**
  Includes the (computed) switching activities for the hierarchical output ports.
-pin

Writes out a pin-based TCF file. This implies that switching activities on all pins are written out.

By default, the RC-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

**Troubleshooting** in Low Power in Encounter RTL Compiler

**TCF Syntax** in *Toggle Count Format Reference*

Related command: read_tcf on page 997

Affected by these attributes: lp_asserted_probability

lp_asserted_toggle_rate
Advanced Low Power Synthesis

- apply_power_intent on page 1018
- check_cpf on page 1021
- check_library on page 1024
- commit_power_intent on page 1030
- create_library_domain on page 1031
- read_power_intent on page 1032
- report_low_power_cells on page 1034
- report_low_power_intent on page 1035
- verify_power_structure on page 1036
- write_power_intent on page 1038
**apply_power_intent**

```plaintext
apply_power_intent
   [-design design -module module]
   [-keep_power_domain_boundaries] [-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.

- **-keep_power_domain_boundaries**
  Prevents optimization beyond the power domain boundaries.
  
  In general, there is no need to preserve power domain boundaries as the isolation cells and level-shifters protect the boundaries. Also RTL Compiler can achieve best QoR.

  In case of non-optimal RTL and very small power domain sizes, the power domain hierarchies can be optimized during synthesis. Although this optimization produces the netlist with the best QoR, and passes the low power equivalence checking, it can result in Conformal Low Power (CLP) violations. These CLP violations usually flag unnecessary level-shifters or isolation cells.

  Use this option, to avoid these CLP violations. However, RTL Compiler will not achieve the best possible QoR in this case.
-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.

rc:/>read_power_intent –module top top.1801
rc:/>elaborate
rc:/>cd des/*
rc:/>designs/top_xyz
rc:/>apply_power_intent -design top_xyz -module top

■ The following example shows a summary report shown in the log when the -summary option is specified.

Summary
=================================================================================================
Power Intent File (format:IEEE-1801) : ls_either_1.upf
=================================================================================================
Number of Power Domains : 4
Number of Isolation Rules : 0
Number of Level Shifter Rules : 2
Number of State Retention Rules : 0
Number of Power Modes : 2
=================================================================================================

=================================================================================================
Partially Supported Commands Summary
=================================================================================================
Type:IEEE-1801 ls_either_1.upf
=================================================================================================
Commands: set_port_attributes
Suggestions: Use only primary I/O ports or pins of macro-model.
=================================================================================================
See 'IEEE-1801 Support in RTL Compiler' in 'Low Power in Encounter RTL Compiler' for more information.
Related Information

See the following chapters in *Low Power in Encounter RTL Compiler*

- Using CPF for Multiple Supply Voltage Designs
- Using CPF for Designs Using Power Shutoff Methodology
- Using CPF for Designs Using Dynamic Voltage Frequency Scaling
- Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology

Related commands:  
  - commit_power_intent on page 1030
  - read_power_intent on page 1032
  - write_power_intent on page 1038
check_cpf

check_cpf
   [-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 Encounter® 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 Encounter Conformal Low Power allowing you to debug any errors further in Conformal Low Power.
- 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.
- 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 Encounter 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.

```
rc:/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.
```

### Related Information

See the following sections in *Design for Test in Encounter RTL Compiler*

- **MSV with DFT Flow**
- **PSO with DFT Flow**

*Interfacing with Conformal Low Power* in *Interfacing between Encounter RTL Compiler and Encounter Conformal*
Common Power Format Rule Checks in *Encounter® Conformal® Low Power Reference Manual*

Affected by this attribute:  

`wclp_lib_statetable`
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
-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.

rc:/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
============================================================
The following command checks the libraries in library domain `lib_120v`.

```bash
rc:/> 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:      closed
```

Unusable libcells

```
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
-----------------------------------------------------------------------------
```
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.

```
rc:/> 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) 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
Technology libraries: ...
Technology libraries: ...
Technology libraries: ...
PPPAD 0.74-0.74 1.2-1.2 up to
Technolog...
The following command checks the function of library cell `LSHLL1CLXL`.

```
rc:/> check_library -libcell LSHLL1CLXL
```

```
Library Libcell Level Isolation Retention Always Function
domain shifter cell flop ON
lib_074v LSHLL1CLXL true true false false false Y = A * B
```

The following command checks the level shifter characteristics of the specified cell.

```
rc:/> 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
```

```
```
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.

```
rc:/> 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_SDFPRQNX10 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_SDFPRQNX10_SRPG
HD65_LS_SDFPRQX10_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**

See the following chapters in *Low Power in Encounter RTL Compiler*

- **Using CPF for Multiple Supply Voltage Designs**
- **Using CPF for Designs Using Power Shutoff Methodology**
- **Using CPF for Designs Using Dynamic Voltage Frequency Scaling**

Related commands: [read_power_intent](#) on page 1032
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

See the following chapters in Low Power in Encounter RTL Compiler

- Using CPF for Multiple Supply Voltage Designs
- Using CPF for Designs Using Power Shutoff Methodology
- Using CPF for Designs Using Dynamic Voltage Frequency Scaling
- Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology

Related commands:  apply_power_intent on page 1018
                  read_power_intent on page 1032
                  write_power_intent on page 1038
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:

  rc:/> 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

Create Library Domains in Encounter RTL Compiler Synthesis Flows.

Related attribute: library
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 UPF format 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

Using CPF for Multiple Supply Voltage Designs

Using CPF for Designs Using Power Shutoff Methodology
Using CPF for Designs Using Dynamic Voltage Frequency Scaling

Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology in Low Power in Encounter RTL Compiler

Related commands: 

- apply_power_intent on page 1018
- commit_power_intent on page 1030
- write_power_intent on page 1038
report low_power_cells

For more information, refer to report low_power_cells in Chapter 9, “Analysis and Report.”
report low_power_intent

For more information, refer to report low_power_intent in Chapter 9, “Analysis and Report.”
verify_power_structure

verify_power_structure
  [-isolation] [-level_shifter] [-retention]
  [-all] [-lp_only]
  [-pre_synthesis | -post_synthesis] [-detail]
  [-pre_read script] [-pre_exit script]
  [-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, RTL Compiler 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 Encounter 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.
- **-detail** Provides a detailed report.
- **-debug** Creates a dofile and other required files for Encounter 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.
- **-level_shifter** Reports only level-shifter related violations.
- **-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.
- **-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.

-run_dir directory  Specifies the directory in which the required files for Encounter Conformal Low Power must be stored.

    Default: .

-tclmode  Specifies to generate the dofile as a Tcl script.

Related Information

See the following sections in Low Power in Encounter RTL Compiler

- **Verify Added Power Logic** in “Using CPF for Multiple Supply Voltage Designs”
- **Verify Added Power Logic** in “Using CPF for Designs Using Power Shutoff Methodology”
- **Verify Added Power Logic** in “Using CPF for Designs Using Dynamic Voltage Frequency Scaling”

See the following sections in Design for Test in Encounter RTL Compiler

- **MSV with DFT Flow**
- **PSO with DFT Flow**

Interfacing with Conformal Low Power in Interfacing between Encounter RTL Compiler and Encounter Conformal

Related command: commit_power_intent on page 1030

Affected by this attribute: wclp_lib_statetable
write_power_intent

write_power_intent [-1801 | -cpf]
  [-base_name string] [-design design]
  [-overwrite] [-to_macro] [-partition_assemble]

Writes out an updated power intent file in the IEEE 1801-2009 standard. The command returns the path to the file with the power intent information.

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 the 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.

For more information on the command and command option support, refer to 1801 Support in RTL Compiler

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.

-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.

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

See the following chapters in Low Power in Encounter RTL Compiler

Using CPF for Multiple Supply Voltage Designs

Using CPF for Designs Using Power Shutoff Methodology

Using CPF for Designs Using Dynamic Voltage Frequency Scaling

CPF Partitioning Flow

Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology

Affected by this attribute: part_power_intent_file

Related commands: apply_power_intent on page 1018
commit_power_intent on page 1030
read_power_intent on page 1032
Design Manipulation

- change_link on page 1043
- change_names on page 1045
- clock_gating on page 1052
- delete_unloaded_undriven on page 1053
- edit_netlist on page 1054
- edit_netlist bitblast_all_ports on page 1056
- edit_netlist bitblast_port on page 1057
- edit_netlist connect on page 1058
- edit_netlist dedicate_subdesign on page 1060
- edit_netlist delete on page 1061
- edit_netlist disconnect on page 1062
- edit_netlist group on page 1064
- edit_netlist hier_connect on page 1066
- edit_netlist new_design on page 1067
- edit_netlist new_instance on page 1068
- edit_netlist new_port_bus on page 1070
- edit_netlist new_primitive on page 1071
- edit_netlist new_subport_bus
- edit_netlist ungroup on page 1074
- edit_netlist uniquify on page 1075
- group
- insert_tiehilo_cells on page 1077
- `mv` on page 1080
- `remove_cdn_loop_breaker` on page 1082
- `reset_design` on page 1084
- `rm` on page 1085
- `ungroup` on page 1086
- `uniquify`
**change_link**

```bash
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.
Examples

- The following command changes the reference of the hierarchical instances `top/A`, `top/B`, and `top/C` to `patch`.

```bash
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`:

```bash
rc:/> 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`.

```bash
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`.

```bash
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`.

```bash
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.

```bash
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
```
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.

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

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.

  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.

-replace_str string

Specifies the replacement string. Specify NULL for a null string.

Default:_

-reserved_words string

Specifies words to be avoided, such as "begin end".
Examples

- The following example adds a suffix _t to the design name:

  rc:/> 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.

  rc:/> change_names -instance -map {{"n" "N"} {"_" "-"}}

- The following example replaces all lowercase “a” with uppercase “A” on all subdesigns and subports.

  rc:/> change_names -map {{"a" "A"}} -subdesign -subport_bus
The following example replaces any instances of `ab`, `bc`, or `ca` with `@` in all object names.

```
rc:/> 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.

```
rc:/> change_names -max_length 12 -subdesign
```

The following example ignores case sensitivity. Because the design contains nets `n_73` and `N_73`, RTL Compiler renames one of the nets to avoid a naming conflict.

```
rc:/> mv n_73 N_73
/designs/alu/nets/N_73
rc:/> mv n 72 n 73
/designs/alu/nets/n_73
rc:/> 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:

```
rc:/designs/test/ports_in> ls
rc:/designs/test/ports_in> change_names -port_bus -map { {"[ " "] " [ "] "} "[ " "] "}
rc:/designs/test/ports_in> ls
```

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".

The following example allows all capital and lower case letters, numbers, underscores, backslashes, and brackets:

```
rc:/> change_names -allowed \
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_\[\]
```

**Note:** You cannot use the dash "-" to indicate inclusion. That is, the following example is not allowed:

```
rc:/> 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:

```
rc:/> 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.

```bash
change_names -name_collision -first_res d -prefix xx
```

The following example shows a part of a netlist that includes VHDL reserved words:

```vhdl
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.

```bash
rc:/> change_names -vhdl
```

Now the netlist does not contain any VHDL keywords:

```vhdl
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:

```bash
rc:/> ls /designs/test/ports_in
rc:/> change_names -lowertoupper
Setting in1[1] --> IN1[1]
...
rc:/> ls /designs/test/ports_in
```

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.

```bash
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`.

```bash
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.

```bash
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 276
clock_gating

Refer to clock_gating in Chapter 12, “Low Power Synthesis.”
delete_unloaded_undriven

delete_unloaded_undriven
   [-disconnect] [-force_bit_blast] [-all]
   [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.

-verbose Enables verbose output.
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 1056
                  edit_netlist bitblast_port on page 1057
edit_netlist connect on page 1058
edit_netlist dedicate_subdesign on page 1060
edit_netlist delete on page 1061
edit_netlist disconnect on page 1062
edit_netlist group on page 1064
edit_netlist hier_connect on page 1066
edit_netlist new_design on page 1067
edit_netlist new_instance on page 1068
edit_netlist new_port_bus on page 1070
edit_netlist new_primitive on page 1071
edit_netlist new_subport_bus on page 1073
edit_netlist uniquify on page 1075
edit_netlist group on page 1064
edit_netlist ungroup on page 1074

Affected by this attribute: ui_respects_preserve
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 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:

...  
rc:/> synthesize  
...  
rc:/> edit_netlist bitblast_all_ports top  
rc:/> ls /designs/top/ports_in

AI_0 AI_1 AI_2 AI_3

Related Information

Affected by this attribute: bit_blasted_port_style
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:
%$_%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
edit_netlist connect

edit_netlist connect
   {constant|pin|pgpin|port|subport}
   {constant|pin|pgpin|port|subport}
   [-net_name string]

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.

-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.
**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`.

```
rc:/designs/alu/ports_in> edit_netlist connect A C
/designs/alu/nets/A_
```

**Related Information**

Related command: `edit_netlist disconnect` on page 1062
edit_netlist dedicate_subdesign

edit_netlist dedicate_subdesign 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.

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.

Example

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:

rc:/> 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.
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.
edit_netlist disconnect

```
edit_netlist disconnect {pin|pgpin|port|subport}
```

Disconnects a single pin, port, or subport from each object it is 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.

**Examples**

- The following example disconnects input port data[4].
  ```
  rc:/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.
Related Information

Using the edit_netlist Command in Design for Test in Encounter RTL Compiler

Related command: edit_netlist connect on page 1058
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.
  rc:/> 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.
  rc:/> 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.
Related Information

Grouping and Ungrouping Objects in Using Encounter RTL Compiler

Related command: edit_netlist ungroup on page 1074
Affected by these attributes: group_generate_portname_from_netname
                          group_instance_suffix
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 output 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**

[Using the edit_netlist Command in Design for Test in Encounter RTL Compiler](#)

Related command:  
    [edit_netlist connect](#) on page 1058
**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`.
  ```bash
  rc:/> 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.
  ```bash
  rc:/> 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 1070
**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:**

```
rc:/> edit_netlist new_instance -name TEST_SUB /designs/TEST 
     /designs/TEST/instances_hier/
rc:/> ls /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.**

```
rc:/> edit_netlist new_instance -name TEST_SUB /designs/TEST -quiet 
     /designs/TEST/instances_hier/TEST_SUB3
```

The name given to the new subdesign in this case is **TEST_SUB3**.
Related Information

Related command: edit_netlist new_subport_bus on page 1073
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:

  rc:/> edit_netlist new_port_bus -name a_in -input

Related Information

Related command: edit_netlist new_design on page 1067
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.

-quiet              Suppresses the warning messages regarding naming
                    conflicts.

Examples

■ The following example creates a new buffer instance called I101 in design DESIGNA:
  rc:// edit_netlist new_primitive -name I101 buf DESIGNA
  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:

```
rc:/> 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.
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:
   rc:/ edit_netlist new_subport_bus -name a_in -input

Related Information

Related command: edit_netlist new_instance on page 1068
**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:

  ```
  rc:/> 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:

  ```
  rc:/designs/test/instances_hier> edit_netlist ungroup -prefix inst1_test \ inst1
  rc:/designs/test/instances_comb> ls
  ```

  ```
  inst1_test_g1/ inst1_test_g2/ inst1_test_g3/
  ```

**Related Informations**

Grouping and Ungrouping Objects in Using Encounter RTL Compiler

Related command: **edit_netlist group** on page 1064
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`:
  ```
  rc:/designs/top/subdesigns> ls
  ./ A/ B/
  ```
  In order to uniquify the subdesigns, issue the `edit_netlist uniquify` command on `top`:
  ```
  rc:/designs/top/subdesigns> edit_netlist uniquify /designs/top
  rc:/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
  }
  ```
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.
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.
-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 1s and 0s 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:

rc:/> 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/ml are skipped. Use the '-all' option to avoid skipping of scan pins.
/designs/ml/instances_seq/foo-bx_reg/pins_in/SE
/designs/ml/instances_seq/tm_reg/pins_in/SE

Done connecting 1'b0 and 1'b1 to TIELO/TIEHI cells
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 UI in module top will be skipped.

```verilog
module UI(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;
  UI inst_U1(.A (a), .B (1'b0), .C (c), .Sel (sel), .Z (z));
endmodule
```

Related Information

Removing Assign Statements in Using Encounter RTL Compiler

Related attributes: ui_respects_preserve use_tiehilo_for_const
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.
- **-slash_ok** Indicates that the destination name can have embedded slashes.
Examples

- The following example changes the name of design `comp` to `comp_test`.
  ```bash
  rc:/designs> ls
  comp
  rc:/designs> mv comp comp_test
  rc:/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:
  ```bash
  rc:/> mv /designs/dpdlalgn/subdesigns/mux muxYYY
  rc:/> ls /designs/dpdlalgn/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:
  ```bash
  rc:/designs/alu/instances_seq> mv aluout_reg_0 aluout_reg_1 -flexible
  /designs/alu/instances_seq/aluout_reg585
  rc:/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 1045
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.

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.

rc:/> 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
rc:/> remove_cdn_loop_breaker -instance [find / -inst inst1/cdn_loop_breaker]
rc:/> report cdn_loop_breaker
============================================================================
   Generated by: Encounter(R) RTL Compiler 8.1.200
   Generated on: Oct 20 2008 03:53:12 PM
   Module:                 loop
   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: \texttt{report cdn_loop_breaker} on page 430
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.

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:

rc:/designs/phoenix/timing/external_delays> ls
in_1    out_2

rc:/designs/phoenix/timing/external_delays> reset_design phoenix
rc:/designs/phoenix/timing/external_delays> ls
./
The command `rm` 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:

```bash
rc:/> 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.

```bash
rc:/> 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'.

rc:/> find . -clock *
I cannot find any clock named * here
Failed on find . -clock *
```
ungroup

-ungroup [ -all | -flatten instance... | -threshold integer | instance...] [-simple] [-prefix string] [-exclude instances] [-only_user_hierarchy] [-force]

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 instance**: 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.
Examples

- The following example ungroups the instance named CRITICAL_GROUP:
  
  ```
  rc:/> ungroup [find / -instance CRITICAL_GROUP]
  ```

- In the following example, every instance under the inst hierarchical instance will be ungrouped except the inst_sub2 instance:
  
  ```
  rc:/designs/ksable_hier/instances_hier/inst/instances_hier> ls
   ./      inst_sub1/  inst_sub2/  inst_sub3
  rc:/designs/ksable_hier/instances_hier/> ungroup inst -flatten -exclude \[find . -instance inst/inst_sub2
  ```

Related Information

Related command: edit_netlist ungroup on page 1074
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.

**Note:** Preserved modules cannot be uniquified.

Alias for edit_netlist uniquify.
Customization

- add_command_help on page 1090
- define_attribute on page 1091
- mesg_make on page 1096
- mesg_send on page 1098
- parse_options on page 1099
add_command_help

add_command_help command_name help category

Adds a short help message for a new command to RTL Compiler's online help system. Examples of add_command_help can be found in the installation lib/cdn/rc 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:

rc:/> proc hello_world {hello_world} { echo "Hello world" }
rc:/> add_command_help "hello_world" "Says hello to the whole world" my_category
rc:/> help hello_world
That command is:

my_category
=============================================================================
hello_world  Says hello to the whole world
Command details:
Hello world
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] [-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.
-data_type string

Defines the data type of the attribute. Possible data types are:

- boolean
- fixed point
- double
- integer
- 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 RTL Compiler prompt.

-obsolete

Specifies whether the defined attribute is obsolete.
Examples

- The following example defines a boolean attribute named new_libpin for a new category named my_libpin:

rc:/> define_attribute -data_type boolean -obj_type libpin -category my_libpin
new_libpin
rc:/> 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.

rc:/> proc check_fxn {obj val} {
   if {$val < 0} {
      return 0
   } else {
      return 1
   }
}
rc:/> 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

-set_function string

Specifies a previously defined Tcl procedure’s name. This option allows you to override user-defined values provided it conforms to the parameters in the Tcl procedure you created. 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)
The following command is a valid use of the newly created `test_check` attribute:

```bash
rc:/> set_attribute test_check 1 /
Setting attribute of root '/': 'test_check' = 1
```

The following command would be an invalid use:

```bash
rc:/> 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):

```bash
rc:/> proc set_fxn {obj new_val cur_val} {
  ==>   if {new_val > cur_val} {
  ==>     return $new_val
  ==>   }
  ==>   return $cur_val
  ==> }
rc:/> 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:

```bash
rc:/> set_attribute test_set 1
Setting attribute of root '/': 'test_set' = 1
```

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`:

```bash
rc:/> set_attribute test_set 2 /
Setting attribute of root '/': 'test_set' = 2
```
The attribute’s value will not be changed in the following example. The value will remain at 2:

```
rc:/> 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:

```
rc:/> proc compute_fxn {obj} {
   =>   return 42
   => }
rc:/> define_attribute test_compute -obj_type root -data_type integer \ 
   -category test -help_string "test compute function" \ 
   -compute_function compute_fxn -computed
```

The `test_compute` value will always be 42, as defined in the Tcl procedure:

```
rc:/> get_attribute test_compute /
42
```

```
rc:/> 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
```
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.
Example

The following example creates a message in the test group and assigns it a unique identification number (501) within that group:

```
rc:/> mesg_make -group test -id 501 -short_desc note_bene \
===> -long_desc "search for lost time" -warnIng
/messages7/test/test-501
rc:/> 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 1098
mesg_send

mesg_send message string

Accesses the various native messages of RTL Compiler 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 1097 for mesg_make:

rc:/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:

rc:/> mesg_send /messages/VHDLPT/VHDLPT-500 "file not found" -caller read_hdl
Error : Cannot open file. [VHDLPT-500] [read_hdl]
: file not found

Related Information

Affected by this command: mesg_make on page 1096
parse_options

parse_options cmd file_var [args] [code var]...

Interfaces to the RTL Compiler internal command option parser. The RTL Compiler 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`.
- RTL Compiler 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 RTL Compiler 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:

```tcl
proc your_procedure {args} {
    ... (code that implements the procedure)
}
```
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

<dirtypes>

d  directory object

e  enum

f  float

<dirtypes>
n  integer number

<dirtypes>
s  string

<y> is a single character indicating whether the option is optional or required

<dirtypes>
o  optional

<y> is a single character indicating whether lists are accepted

<dirtypes>
m  Accepts multiple values: lists OK

<s> Accepts single value only: no lists
<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:

  rc:/> parse_options hello_world file_var {> ./tmp} \
  1
  rc:/> puts $file_var "Hello world!"
  rc:/> close $file_var

- The following example shows a required argument with the -design flag that must be the name of a subdesign (block).

  rc:/> parse_options hello_world file_var {-design addinc65} \ 
  ==> "-design drs(subdesign) A module" my_design \
  1
  rc:/> 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.

  rc:/> set level 300 
  300
  rc:/> parse_options hello_world file_var {-t addinc65} \ 
  ==> "-top Bos Do the top`level thing" top \
  1
The following example shows how online help works:

```bash
rc:/> 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
Applets

- Introduction on page 1104
- applet on page 1105
- applet avail on page 1106
- applet install on page 1108
- applet list on page 1109
- applet load on page 1110
- applet set defaults on page 1112
- applet update on page 1113
- applet version on page 1114
- applet whatis on page 1115
Introduction

Applets are **non-productized** scripts that can be used for a variety of purposes, such as report generation, template management, text histogram generation, testcase 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.
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.
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

-detailed
   Reports the applets installed per installation: the list for your local installation, and the list available on the server.

-location applet_installation
   Specifities 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.

rc:/> applet avail

...

Info: Collecting applet server information...

Applets Local/Server information:
   Local Install (/home/me/.localApps/rc)
   Remote Server <install>/tools.<platform>/lib/applets

<table>
<thead>
<tr>
<th>Name</th>
<th>Installed</th>
<th>Server Ver.</th>
<th>Summary Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compare_power</td>
<td>6</td>
<td>1.15</td>
<td>Generate HTML comparison report of power metrics..</td>
</tr>
<tr>
<td>create_tcase</td>
<td>5</td>
<td>1.42</td>
<td>generate / load / save testcases</td>
</tr>
<tr>
<td>Hermes</td>
<td>3</td>
<td>1.14</td>
<td>Hermes Design Assistant</td>
</tr>
<tr>
<td>lock</td>
<td>5</td>
<td>1.12</td>
<td>Guarantee atomic operations using a LOCKFILE...</td>
</tr>
</tbody>
</table>
The following command lists the outdated applets.

```
rc:/> applet avail -outdated
===========================================================================
...                                                                                       
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>'

The following command lists the local applets.

```
rc:/> applet avail -local
===========================================================================
...                                                                                       
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_tcage | 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>'
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.

rc:/> applet install -location ~/rc_ex/applets compare_gates
Info: Collecting applet server information...
Installing applet 'compare_gates'...
Updating applet catalog information...
....
rc:/> 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
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.

```bash
rc:/> applet load compare_power
...
rc:/> applet list
===========================================================================
...
===========================================================================

The following applets have been loaded:
    compare_power
genenerate_report
time_info

rc:/>
```
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 (RTL Compiler 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.
Example

The following command loads the compare_power applet together with all its dependencies.

```bash
rc:/> 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/time_info/time_info.tcl' (Tue Aug 23 15:52:34 -0700 2011)...
Loading applet 'compare_power'...
rc:/>
applet set_defaults

applet set_defaults

Sets the applet_mode, applet_server, and applet_search_path attributes to their defaults.
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

```
rc:/> applet update
Info: Collecting applet server information...
Info: Updating applet installation /home/mynname/.localApps/rc...
Updating applet 'time_info' (1.1 -> 1.2)...
Info: Connecting to server....
Info: Collecting applet server information...
```
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.
  
  rc:/ applet version
  applet command Version: 4

- The following command returns the versions of the time_info applet.
  
  rc:/ applet version time_info
  applet 'time_info' Version: 1.52
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.
  
  rc:/> applet whatis {time_info lock}
  
  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.
  
  rc:/> 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.
Index

A

abstract model
  logic, writing 276
  scan, reading 875
  scan, writing 918
adding
  design 1067
  directory to stack 59
  help for user command 1090
  user-defined command 1099
  user-defined message 1096
area, reporting 425
assigning, paths to cost group 346
ATPG file, writing out 642, 836, 904
Attribute
  defining new
    defining custom 1091
attributes
  retrieving value 82
  setting value 109

B

bitblasting
  one port 1054, 1057
  bitblasting, all ports 1056
body segment, defining 739
boundary scan logic
  previewing changes 790
busses
  renaming 1045

C

cells
  reporting cell count 425
  reporting technology library cells used 470
changing
  attribute value
    defined by RC 109
    directory in design hierarchy 39
  names
component 190
instance of 1068
library domains (MSV) 1031
parameter for Chipware component 196
pin 198
port bus 1070
subdesign 1060
subport bus 1073
synthetic operator 193
timing constraints for instance 324
user-defined message 1096
critical path, reporting timing slack for 541
in design hierarchy 60
in UNIX 99

D
defining
boundary scan segment 695
clock 318
configuration mode 699
cost group 323
DFT test clock 719, 762
input and output delays 326
multi-cycle path 332
paths for timing constraints 351
scan clock for LSSD 745, 748
shift-enable signal for DFT 751
test-mode signal for DFT 766
user_defined scan chain 739
user-defined abstract segment 689
design
area 425
creating
generic netlist 380
new design 1067
incremental optimization 381
instantiating 1068
mapping 379
removing 1080
renaming 1045
synthesizing 377
uniquifying 1075, 1088
design hierarchy
adding hierarchy level 1064, 1076
changing directories in 39
finding object type 46
listing information 52, 53

E
editing
script 87
UNIX shell command 111, 112
execiting, from RTL Compiler 81

F
fanin, reporting 413
fanout, reporting 417
filtering, objects  43
find, object type  46
finding
    corresponding input or output  51
    object base name  38
    object directory name  41
    object type  46
    type of object  64
fixing, DFT rule violations  772
flattening, instances  1086
flip-flops
    checking for DFT rule violations  648
    reporting scannability  458
Floorplan (DEF)
    Reading  602
    Writing  624
G
generic netlist
    creating  380
GUI
    update  129
H
HDL files, reading  216
HDL Viewer
    remove data  131
head segment, defining  741
help
    adding for new command  1090
    providing for commands  69, 86
highlight
    clear  134, 148
    color  135, 157
I
incremental optimization  381
input delay, defining  326
inserting
    clock-gating logic in mapped netlist  981
    shadow logic  846
    test point  852
    user-defined test point  858
    wrapper cell  861
instances
    grouping  1064, 1076
    removing from design  1080
    renaming  1045
    ungrouping  1074
    isolation rule
    removing  1080
L
leakage power, reporting  507
length
    abstract segment  690
    maximum chain length  742
level-shifter groups
    removing  1080
library cell, instantiating  1068
library domains
    creating  1031
licenses
    checked out, listing  94
    checking out  91
listing
    object information  52, 53
    lookup element type, chain-specific  743
logic abstract model, definition  276
logic abstract, writing  276
loop breakers
    removing  1082
    reporting  430
M
mapped netlist
    creating  380
    writing out  276
mapping
    design  377
    non-scan flops with scan-equivalent  884
memory resources, reporting  486
messages
    creating customized  1096
    reporting  488
    sending  1098
    suppressing  115
mode, specifying for power, timing analysis, and optimization  315
MSV design
    library domains, creating  1031
multi-cycle path, defining 332

N
names
  adding prefix 1047
  limiting length 1047
  mapping characters 1047
net power, reporting 507
nets
  renaming 1045

O
object
  finding base name 38
  finding directory name 41
object path
  return a selection list for instance, net, pin, and port objects 128
object type, finding 46
object types
  listing all attributes for a type 82, 109
objects
  connecting 1058
  disconnecting 1062
  filtering on attribute value 43
  instantiating 1068
  listing information for an object 52, 53
  removing from design hierarchy 1054, 1061, 1085
  renaming 1080
  returning type of an object 64
  selecting in design hierarchy 43
observability logic for clock-gating logic
  inserting 982
OPCG logic
  previewing changes 825
OPCG segments
  previewing connections 679
optimizing
  for area or timing 371
  incrementally 378
  RTL 378
output delay, defining 326
output, redirecting 104

P
parameterized module, elaborating 360
path
  return path for a selected object 128
path constraints, modifying 336
paths
  assigning to cost group 346
  creating for timing exception 351
  unconstraining 343
pins
  connecting manually 1058
  disconnecting 1062
port buses
  creating new object 1070
  removing from design 1080
  renaming 1045
ports
  bitblasting 1056
  connecting manually 1059
  disconnecting 1062
  renaming 1045
power
  reporting 507
  sorting report 509
power domain
  removing 1080
primitive cell, creating 1071
probability values of nets
  reading or updating 997
  writing to SAIF file 1013
  writing to TCF file 1015

R
reading
  HDL files 216
  SAIF file 992
  SDC constraints 227
  TCF file 997
remove
  data 131
removing
  clock-gating logic 986
  directory from stack 58
  hierarchy from design 1074
  object from design hierarchy 1061, 1085
  renaming
<table>
<thead>
<tr>
<th>instances of object type</th>
<th>1045</th>
<th>object</th>
<th>1080</th>
</tr>
</thead>
<tbody>
<tr>
<td>reporting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>area of design</td>
<td>425</td>
<td>cell types, used</td>
<td>470</td>
</tr>
<tr>
<td>clock information of design</td>
<td>438</td>
<td>clock-gating information</td>
<td>432</td>
</tr>
<tr>
<td>design rule violations</td>
<td>446</td>
<td>DFT registers</td>
<td>458</td>
</tr>
<tr>
<td>DFT setup information</td>
<td>462</td>
<td>DFT violations</td>
<td>466</td>
</tr>
<tr>
<td>inferred datapath operators</td>
<td>441</td>
<td>instance information</td>
<td>476</td>
</tr>
<tr>
<td>memory resources</td>
<td>486</td>
<td>messages in current session</td>
<td>488</td>
</tr>
<tr>
<td>net information</td>
<td>499</td>
<td>port information</td>
<td>505</td>
</tr>
<tr>
<td>scan chains</td>
<td>448</td>
<td>timing information</td>
<td>541</td>
</tr>
<tr>
<td>retrieving</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>attribute value</td>
<td></td>
<td>defined by RC</td>
<td>82</td>
</tr>
<tr>
<td>clock ports</td>
<td>409</td>
<td>fanin information</td>
<td>413</td>
</tr>
<tr>
<td>fanout information</td>
<td>417</td>
<td>input ports</td>
<td>71</td>
</tr>
<tr>
<td>object information</td>
<td>52, 53</td>
<td>output ports</td>
<td>72</td>
</tr>
<tr>
<td>UNIX working directory</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTL Compiler</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exiting</td>
<td>81</td>
<td>quit</td>
<td>103</td>
</tr>
<tr>
<td>resuming process</td>
<td>116</td>
<td>suspending process</td>
<td>116</td>
</tr>
<tr>
<td>RTL optimization</td>
<td>378</td>
<td>RTL power analysis, enabling</td>
<td>509</td>
</tr>
</tbody>
</table>

S

SAIF file
- reading | 992 |
- writing | 1011 |

scan abstract model, creating | 918 |
scan abstract model, reading | 875 |
scan chain
- tool-created
  - test clock domain associated with name | 704, 706, 736, 754, 780 |
scan-chain configuration
- previewing | 684 |

reporting | 448 |
scan chains
- connecting | 681 |
- defining | 739 |
- mixing edges of same clock on | 896 |
- removing | 1080 |
- reporting | 448 |
scan data input
- creating | 740 |
- specifying for chain | 742 |
- specifying for scan segment | 691, 737 |
scan data output
- creating | 740 |
- specifying for chain | 742 |
- specifying for scan segment | 691, 737 |
- using existing port | 671, 743, 768, 801 |
scan segment
- defining | 689, 704, 706, 736, 754, 780 |
- removing | 1080 |
- shift enable
  - confirming if connected | 690, 737 |
  - using in scan chain
    - as body segment | 739 |
    - as head segment | 741 |
    - as tail segment | 743 |
scanDEF file, writing out | 968 |
scripts
- executing | 87 |
- writing out | 289 |
SDC constraints
- reading in | 227 |
- unsupported constructs | 227 |
- writing out | 292 |
shadow logic
- inserting | 846 |
- previewing changes | 848 |
shift-enable port
- for scan segment | 691 |
  - confirming if connected | 690, 737 |
- for shift-enable signal | 731, 753 |
shift-enable signal
- defining | 751, 766 |
- specifying default | 751 |
- specifying for chain | 743 |
slack, reporting at timing endpoints | 542 |
SPEF
- Reading | 608 |
- Writing | 626 |
subdesigns
- creating by copying existing subdesign | 1060 |
instantiating 1068
renaming 1045
uniquifying 1075, 1088
subport buses
creating new object 1073
removing from design 1080
renaming 1045
subports
connecting manually 1059
disconnecting 1062
switching power, reporting 507
synthesis
creating generic netlist 380
incremental optimization 381
mapping 380

T
tail segment, defining 743
TCF file
reading 997
writing 1015
test mode signal, defining 766
test point
inserting 852
possible types 854
timing constraints
deriving for instance 324
reporting violations on 541
writing out 289
timing exception, created by
modifying path constraints 336
overriding default clock edge relationship 332
specifying timing constraints 340
unconstraining paths 343
timing units 541
toggle counts of nets
reading or updating 997
writing to SAIF file 1013
writing to TCF file 1015

U
uniquifying, design or subdesign 1075, 1088
UNIX shell command
executing in RTL Compiler 111, 112
user-defined test point, inserting 858

utilization map
create snapshot for documentation 166

W
wireload model, reporting 425
worst paths, reporting timing for 544
wrapper cell, inserting 861
writing
compression model 910
DFT abstract 918
logic abstract 276
netlist file 276
SAIF file 1011, 1013
scanDEF file 968
SDC constraints 292
TCF file 1015
timing and design rule constraints 289