

# **Command Reference for Encounter® RTL Compiler**

**Product Version 14.2**  
**October 2014**

© 2003-2014 Cadence Design Systems, Inc. All rights reserved.  
Printed in the United States of America.

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

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

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

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

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

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

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

# Contents

---

<u>Alphabetical List of Commands</u> .....	21
<u>Preface</u> .....	27
<u>About This Manual</u> .....	28
<u>Additional References</u> .....	28
<u>How to Use the Documentation Set</u> .....	29
<u>Reporting Problems or Errors in Manuals</u> .....	30
<u>Customer Support</u> .....	31
<u>Cadence Online Support</u> .....	31
<u>Other Support Offerings</u> .....	31
<u>Messages</u> .....	32
<u>Man Pages</u> .....	33
<u>Command-Line Help</u> .....	34
<u>Getting the Syntax for a Command</u> .....	34
<u>Getting the Syntax for an Attribute</u> .....	34
<u>Searching for Attributes</u> .....	35
<u>Searching For Commands When You Are Unsure of the Name</u> .....	35
<u>Documentation Conventions</u> .....	36
<u>Text Command Syntax</u> .....	36
<u>1</u>	
<u>Navigation</u> .....	37
<u>basename</u> .....	38
<u>cd</u> .....	39
<u>dirname</u> .....	41
<u>dirs</u> .....	42
<u>filter</u> .....	43
<u>find</u> .....	46
<u>inout_mate</u> .....	51
<u>ll</u> .....	52

## Command Reference for Encounter RTL Compiler

---

<u>ls</u>	53
<u>popd</u>	58
<u>pushd</u>	60
<u>pwd</u>	61
<u>vdir_lsearch</u>	62
<u>vname</u>	64
<u>what_is</u>	65
<u>what_is_list</u>	66
 <u>2</u>	
<u>General</u>	69
<u>? </u>	71
<u>alias</u>	72
<u>all_inputs</u>	73
<u>all_outputs</u>	74
<u>apropos</u>	75
<u>attribute_exists</u>	76
<u>clear</u>	77
<u>clear_redline_terminal</u>	78
<u>date</u>	79
<u>enable_transparent_latches</u>	80
<u>exec_embedded_script</u>	81
<u>exit</u>	83
<u>get_attribute</u>	84
<u>get_liberty_attribute</u>	86
<u>get_read_files</u>	87
<u>help</u>	88
<u>include</u>	89
<u>lcd</u>	90
<u>license</u>	91
<u>license_checkin</u>	92
<u>license_checkout</u>	93
<u>license_feature</u>	95
<u>license_list</u>	96
<u>license_version</u>	97

## Command Reference for Encounter RTL Compiler

---

<u>ls</u>	98
<u>lpopd</u>	99
<u>lpushd</u>	100
<u>lpwd</u>	101
<u>man</u>	102
<u>more</u>	103
<u>quit</u>	105
<u>redirect</u>	106
<u>reset_attribute</u>	108
<u>resume</u>	109
<u>sdc_shell</u>	110
<u>set_attribute</u>	111
<u>sh</u>	113
<u>shell</u>	114
<u>stop_suspend</u>	115
<u>string_representation</u>	116
<u>suppress_messages</u>	117
<u>suspend</u>	118
<u>tcl_load</u>	119
<u>test_super_thread_servers</u>	122
<u>unsuppress_messages</u>	123
 <u>3</u>	
<u>GUI Text</u>	125
<u>General GUI Text Commands</u>	126
<u>gui_balloon_info</u>	127
<u>gui_hide</u>	127
<u>gui_info</u>	127
<u>gui_legend</u>	128
<u>gui_raise</u>	129
<u>gui_reset</u>	129
<u>gui_resume</u>	130
<u>gui_selection</u>	130
<u>gui_show</u>	130
<u>gui_status</u>	131

## Command Reference for Encounter RTL Compiler

---

<u>gui_suspend</u>	131
<u>gui_update</u>	131
<u>HDL Viewer GUI Text Commands</u>	132
<u>gui_hv_clear</u>	133
<u>gui_hv_get_file</u>	133
<u>gui_hv_load_file</u>	133
<u>gui_hv_set_indicators</u>	134
<u>Schematic Viewer GUI Text Commands</u>	135
<u>gui_sv_clear</u>	136
<u>gui_sv_cone</u>	136
<u>gui_sv_get_instance</u>	136
<u>gui_sv_grey</u>	136
<u>gui_sv_highlight</u>	137
<u>gui_sv_load</u>	138
<u>gui_sv_snapshot</u>	139
<u>gui_sv_toolbar_button</u>	139
<u>Physical Viewer GUI Text Commands</u>	140
<u>gui_pv_airline_add</u>	142
<u>gui_pv_airline_add_custom</u>	144
<u>gui_pv_airline_delete</u>	145
<u>gui_pv_airline_display</u>	146
<u>gui_pv_airline_raw_add</u>	147
<u>gui_pv_airline_raw_add_custom</u>	148
<u>gui_pv_align_instance_to_boundary</u>	149
<u>gui_pv_align_instances</u>	150
<u>gui_pv_clear</u>	150
<u>gui_pv_connectivity_airlines</u>	151
<u>gui_pv_deselect</u>	152
<u>gui_pv_display_collection</u>	153
<u>gui_pv_draw_box</u>	154
<u>gui_pv_draw_circle</u>	155
<u>gui_pv_draw_line</u>	156
<u>gui_pv_draw_triangle</u>	157
<u>gui_pv_get_design</u>	158
<u>gui_pv_highlight</u>	159
<u>gui_pv_highlight_hier_instances</u>	161

## Command Reference for Encounter RTL Compiler

---

<u>gui_pv_highlight_update</u>	162
<u>gui_pv_label</u>	163
<u>gui_pv_lineup_instances</u>	164
<u>gui_pv_new_viewer</u>	164
<u>gui_pv_preferences</u>	165
<u>gui_pv_redraw</u>	166
<u>gui_pv_select</u>	166
<u>gui_pv_selection</u>	167
<u>gui_pv_snapshot</u>	168
<u>gui_pv_steiner_tree</u>	170
<u>gui_pv_toolbar_button</u>	171
<u>gui_pv_update</u>	172
<u>gui_pv_zoom_box</u>	173
<u>gui_pv_zoom_fit</u>	173
<u>gui_pv_zoom_in</u>	174
<u>gui_pv_zoom_out</u>	174
<u>gui_pv_zoom_to</u>	175

## 4

<u>Chipware Developer</u>	177
<u>cwd</u>	178
<u>cwd_check</u>	179
<u>cwd_create_check</u>	183
<u>cwd_report_check</u>	185
<u>hdl_create</u>	187
<u>hdl_create_binding</u>	188
<u>hdl_create_component</u>	190
<u>hdl_create_implementation</u>	192
<u>hdl_create_library</u>	194
<u>hdl_create_operator</u>	195
<u>hdl_create_package</u>	196
<u>hdl_create_parameter</u>	198
<u>hdl_create_pin</u>	200

### 5

<u><a href="#">Input and Output</a></u> . . . . .	203
<u><a href="#">decrypt</a></u> . . . . .	206
<u><a href="#">encrypt</a></u> . . . . .	207
<u><a href="#">export critical endpoints</a></u> . . . . .	210
<u><a href="#">read_db</a></u> . . . . .	212
<u><a href="#">read_def</a></u> . . . . .	213
<u><a href="#">read_dfm</a></u> . . . . .	214
<u><a href="#">read_dft_abstract_model</a></u> . . . . .	216
<u><a href="#">read_encounter</a></u> . . . . .	217
<u><a href="#">read_hdl</a></u> . . . . .	218
<u><a href="#">read_io_speclist</a></u> . . . . .	222
<u><a href="#">read_memory_view</a></u> . . . . .	223
<u><a href="#">read_netlist</a></u> . . . . .	224
<u><a href="#">read_pmbist_interface_files</a></u> . . . . .	226
<u><a href="#">read_power_intent</a></u> . . . . .	227
<u><a href="#">read_saif</a></u> . . . . .	228
<u><a href="#">read_sdc</a></u> . . . . .	229
<u><a href="#">read_spf</a></u> . . . . .	231
<u><a href="#">read_tcf</a></u> . . . . .	232
<u><a href="#">read_vcd</a></u> . . . . .	233
<u><a href="#">restore_design</a></u> . . . . .	234
<u><a href="#">split_db</a></u> . . . . .	236
<u><a href="#">write_atpg</a></u> . . . . .	237
<u><a href="#">write_bsdl</a></u> . . . . .	238
<u><a href="#">write_compression_macro</a></u> . . . . .	239
<u><a href="#">write_db</a></u> . . . . .	240
<u><a href="#">write_def</a></u> . . . . .	242
<u><a href="#">write_design</a></u> . . . . .	243
<u><a href="#">write_dft_abstract_model</a></u> . . . . .	245
<u><a href="#">write_dft_rtl_model</a></u> . . . . .	246
<u><a href="#">write_do_ccd</a></u> . . . . .	247
<u><a href="#">write_do_ccd_cdc</a></u> . . . . .	248
<u><a href="#">write_do_ccd_compare_sdc</a></u> . . . . .	249
<u><a href="#">write_do_ccd_generate</a></u> . . . . .	251

## Command Reference for Encounter RTL Compiler

---

<u>write do_ccd propagate</u>	253
<u>write do_ccd validate</u>	255
<u>write do_clp</u>	256
<u>write do_lec</u>	259
<u>write do_verify_cdc</u>	263
<u>write encounter</u>	265
<u>write et_atpg</u>	268
<u>write et_bsv</u>	269
<u>write et_dfa</u>	270
<u>write et_lbist</u>	271
<u>write et_mbist</u>	272
<u>write et_no_tp_file</u>	273
<u>write et_rrfa</u>	274
<u>write_ets</u>	275
<u>write_ett</u>	276
<u>write_forward_saif</u>	277
<u>write_hdl</u>	278
<u>write_io_speclist</u>	282
<u>write_ldb</u>	283
<u>write_logic_bist_macro</u>	284
<u>write_mbist_testbench</u>	285
<u>write_pmbist_interface_files</u>	286
<u>write_pmbist_testbench</u>	287
<u>write_power_intent</u>	288
<u>write_saif</u>	289
<u>write_scandef</u>	290
<u>write_script</u>	291
<u>write_sdc</u>	294
<u>write_sdf</u>	298
<u>write_set_load</u>	303
<u>write_spf</u>	304
<u>write_sv_wrapper</u>	305
<u>write_tcf</u>	308
<u>write_template</u>	309

### 6

<u>Constraints</u> .....	313
<u>clock_uncertainty</u> .....	314
<u>create_mode</u> .....	317
<u>define_clock</u> .....	320
<u>define_cost_group</u> .....	325
<u>derive_environment</u> .....	326
<u>external_delay</u> .....	328
<u>generate_constraints</u> .....	332
<u>multi_cycle</u> .....	334
<u>path_adjust</u> .....	338
<u>path_delay</u> .....	342
<u>path_disable</u> .....	345
<u>path_group</u> .....	348
<u>propagate_constraints</u> .....	351
<u>specify_paths</u> .....	353
<u>validate_constraints</u> .....	359

### 7

<u>Elaboration and Synthesis</u> .....	361
<u>elaborate</u> .....	362
<u>get_remove_assign_options</u> .....	365
<u>merge_to_multibit_cells</u> .....	366
<u>remap_to_dedicated_clock_library</u> .....	367
<u>remove_assigns_without_optimization</u> .....	368
<u>remove_clock_reconvergence</u> .....	371
<u>remove_inserted_sync_enable_logic</u> .....	372
<u>retime</u> .....	373
<u>set_remove_assign_options</u> .....	375
<u>synthesize</u> .....	379

### 8

<u>Hierarchical Flow</u> .....	385
<u>assemble_design</u> .....	386

<u>generate_ilm</u>	387
<u>read_ilm</u>	389
<b>9</b>	
<b><u>Analysis and Report</u></b>	391
<u>all_connected</u>	394
<u>all_des</u>	395
<u>all_des_inps</u>	396
<u>all_des_insts</u>	397
<u>all_des_outs</u>	398
<u>all_des_seqs</u>	399
<u>all_lib</u>	401
<u>all_lib_bufs</u>	402
<u>all_lib_ties</u>	403
<u>analyze_library_corners</u>	404
<u>check_design</u>	406
<u>clock_ports</u>	411
<u>compare_sdc</u>	412
<u>create_timing_bin</u>	413
<u>fanin</u>	415
<u>fanout</u>	419
<u>report</u>	422
<u>report_area</u>	427
<u>report_boundary_opto</u>	429
<u>report_case_analysis</u>	431
<u>report_cdn_loop_breaker</u>	432
<u>report_cell_delay_calculation</u>	433
<u>report_clock_gating</u>	434
<u>report_clocks</u>	440
<u>report_congestion</u>	442
<u>report_datapath</u>	443
<u>report_design_rules</u>	448
<u>report_dft_chains</u>	449
<u>report_dft_clock_domain_info</u>	454
<u>report_dft_core_wrapper</u>	455

## Command Reference for Encounter RTL Compiler

---

<u>report dft_registers</u>	459
<u>report dft_setup</u>	463
<u>report dft_violations</u>	467
<u>report disabled transparent latches</u>	470
<u>report gates</u>	471
<u>report hierarchy</u>	475
<u>report instance</u>	477
<u>report low_power_cells</u>	480
<u>report low_power_intent</u>	483
<u>report memory</u>	487
<u>report memory_cells</u>	488
<u>report messages</u>	489
<u>report mode</u>	491
<u>report multibit_inferencing</u>	492
<u>report net_cap_calculation</u>	496
<u>report net_delay_calculation</u>	497
<u>report net_res_calculation</u>	498
<u>report nets</u>	499
<u>report opcg_equivalents</u>	502
<u>report ple</u>	503
<u>report port</u>	505
<u>report power</u>	507
<u>report power_domain</u>	519
<u>report proto</u>	521
<u>report qor</u>	523
<u>report scan_compressibility</u>	529
<u>report sequential</u>	531
<u>report slew_calculation</u>	534
<u>report summary</u>	535
<u>report test_power</u>	537
<u>report timing</u>	541
<u>report units</u>	552
<u>report utilization</u>	553
<u>report yield</u>	554
<u>statistics</u>	555
<u>statistics add_metric</u>	557

## Command Reference for Encounter RTL Compiler

---

<u>statistics log</u>	558
<u>statistics read</u>	560
<u>statistics remove_metric</u>	561
<u>statistics report</u>	562
<u>statistics reset</u>	565
<u>statistics run_stage_ids</u>	566
<u>statistics write</u>	567
<u>timestat</u>	568
<u>validate_timing</u>	569
 <u>10</u>	
<u>Physical</u>	571
<u>check_floorplan</u>	573
<u>check_placement</u>	577
<u>create_group</u>	579
<u>create_placement_blockage</u>	580
<u>create_placement_halo_blockage</u>	581
<u>create_region</u>	582
<u>create_row</u>	584
<u>create_routing_blockage</u>	585
<u>create_routing_halo_blockage</u>	587
<u>create_track</u>	588
<u>def_move</u>	589
<u>duplicate_register</u>	590
<u>generate_ple_model</u>	592
<u>generate_reports</u>	594
<u>modify_power_domain_attr</u>	596
<u>move_blockage</u>	598
<u>move_instance</u>	599
<u>move_port</u>	600
<u>move_region</u>	601
<u>read_def</u>	602
<u>read_encounter</u>	606
<u>read_sdp_file</u>	607
<u>read_spef</u>	608

<u>report congestion</u>	609
<u>report utilization</u>	610
<u>resize blockage</u>	611
<u>resize region</u>	612
<u>restore congestion map</u>	613
<u>save congestion map</u>	614
<u>specify cell pad</u>	615
<u>specify floorplan</u>	616
<u>summary table</u>	618
<u>update congestion map</u>	620
<u>update gcell congestion</u>	621
<u>update gcell pin density</u>	622
<u>update gcell utilization</u>	623
<u>write_def</u>	624
<u>write_sdp_file</u>	625
<u>write_spf</u>	626

## 11

<u>Design for Test</u>	627
<u>add_opcg_hold_mux</u>	632
<u>analyze_scan_compressibility</u>	633
<u>analyze_testability</u>	642
<u>check_atpg_rules</u>	645
<u>check_dft_pad_configuration</u>	647
<u>check_dft_rules</u>	648
<u>check_mbist_rules</u>	654
<u>compress_block_level_chains</u>	657
<u>compress_scan_chains</u>	660
<u>concat_scan_chains</u>	675
<u>configure_pad_dft</u>	677
<u>connect_compression_clocks</u>	678
<u>connect_opcg_segments</u>	679
<u>connect_scan_chains</u>	681
<u>define_dft</u>	686
<u>define_dft_abstract_segment</u>	689

## Command Reference for Encounter RTL Compiler

---

<u>define_dft_boundary_scan_segment</u>	695
<u>define_dft_dft_configuration_mode</u>	699
<u>define_dft_domain_macro_parameters</u>	702
<u>define_dft_fixed_segment</u>	704
<u>define_dft_floating_segment</u>	706
<u>define_dft_jtag_instruction</u>	708
<u>define_dft_jtag_instruction_register</u>	712
<u>define_dft_jtag_macro</u>	714
<u>define_dft_mbist_clock</u>	719
<u>define_dft_mbist_direct_access</u>	722
<u>define_dft_opcg_domain</u>	725
<u>define_dft_opcg_mode</u>	728
<u>define_dft_opcg_trigger</u>	730
<u>define_dft_osc_source</u>	732
<u>define_dft_pmbist_direct_access</u>	734
<u>define_dft_preserved_segment</u>	736
<u>define_dft_scan_chain</u>	739
<u>define_dft_scan_clock_a</u>	745
<u>define_dft_scan_clock_b</u>	748
<u>define_dft_shift_enable</u>	751
<u>define_dft_shift_register_segment</u>	754
<u>define_dft_tap_port</u>	756
<u>define_dft_test_bus_port</u>	758
<u>define_dft_test_clock</u>	761
<u>define_dft_test_mode</u>	765
<u>dft_trace_back</u>	769
<u>fix_dft_violations</u>	771
<u>fix_scan_path_inversions</u>	775
<u>identify_domain_crossing_pins_for_cgic_and_scan_abstracts</u>	776
<u>identify_multibit_cell_abstract_scan_segments</u>	777
<u>identify_shift_register_scan_segments</u>	779
<u>identify_test_mode_registers</u>	781
<u>insert_dft</u>	784
<u>insert_dft_boundary_scan</u>	787
<u>insert_dft_compression_logic</u>	791
<u>insert_dft_dfa_test_points</u>	802

## Command Reference for Encounter RTL Compiler

---

<u>insert_dft_jtag_macro</u>	806
<u>insert_dft_lockup_element</u>	810
<u>insert_dft_logic_bist</u>	811
<u>insert_dft_mbist</u>	817
<u>insert_dft_opcg</u>	823
<u>insert_dft_pmbist</u>	825
<u>insert_dft_ptam</u>	830
<u>insert_dft_rrfa_test_points</u>	833
<u>insert_dft_scan_power_gating</u>	840
<u>insert_dft_shadow_logic</u>	843
<u>insert_dft_shift_register_test_points</u>	848
<u>insert_dft_test_point</u>	849
<u>insert_dft_user_test_point</u>	855
<u>insert_dft_wrapper_cell</u>	857
<u>insert_dft_wrapper_instruction_register</u>	863
<u>insert_dft_wrapper_mode_decode_block</u>	865
<u>insert_test_compression</u>	867
<u>map_mbist_cgc_to_cgic</u>	870
<u>read_dft_abstract_model</u>	871
<u>read_io_speclist</u>	874
<u>read_memory_view</u>	875
<u>read_pmbist_interface_files</u>	877
<u>replace_opcg_scan</u>	878
<u>replace_scan</u>	880
<u>report_dft_chains</u>	881
<u>report_dft_clock_domain_info</u>	882
<u>report_dft_core_wrapper</u>	883
<u>report_dft_registers</u>	884
<u>report_dft_setup</u>	885
<u>report_dft_violations</u>	886
<u>report_opcg_equivalents</u>	887
<u>report_scan_compressibility</u>	888
<u>report_test_power</u>	889
<u>reset_opcg_equivalent</u>	890
<u>reset_scan_equivalent</u>	891
<u>set_compatible_test_clocks</u>	892

## Command Reference for Encounter RTL Compiler

---

<u>set opcg equivalent</u>	894
<u>set scan equivalent</u>	896
<u>update scan chains</u>	898
<u>write atpg</u>	900
<u>write bsdl</u>	903
<u>write compression macro</u>	906
<u>write dft abstract model</u>	914
<u>write dft rtl model</u>	917
<u>write et atpg</u>	918
<u>write et bsv</u>	926
<u>write et dfa</u>	930
<u>write et lbist</u>	934
<u>write et mbist</u>	938
<u>write et no_tp file</u>	943
<u>write et rrfa</u>	944
<u>write io speclist</u>	948
<u>write logic bist macro</u>	950
<u>write mbist testbench</u>	954
<u>write pmbist interface files</u>	958
<u>write pmbist testbench</u>	960
<u>write scandef</u>	964

## 12

<u>Low Power Synthesis</u>	967
<u>build rtl power models</u>	969
<u>clock gating</u>	971
<u>clock gating connect test</u>	973
<u>clock gating declone</u>	974
<u>clock gating import</u>	975
<u>clock gating insert_in_netlist</u>	977
<u>clock gating insert obs</u>	978
<u>clock gating join</u>	980
<u>clock gating remove</u>	982
<u>clock gating share</u>	984
<u>clock gating split</u>	986

<u>read_saif</u>	988
<u>read_tcf</u>	993
<u>read_vcd</u>	998
<u>report_clock_gating</u>	1002
<u>report_operand_isolation</u>	1003
<u>report_power</u>	1004
<u>state_retention</u>	1005
<u>state_retention_connect_power_gating_pins</u>	1006
<u>state_retention_swap</u>	1007
<u>write_forward_saif</u>	1008
<u>write_saif</u>	1010
<u>write_tcf</u>	1012
<b>13</b>	
<b><u>Advanced Low Power Synthesis</u></b>	1015
<u>apply_power_intent</u>	1016
<u>check_cpf</u>	1019
<u>check_library</u>	1022
<u>commit_power_intent</u>	1028
<u>create_library_domain</u>	1029
<u>read_power_intent</u>	1030
<u>report_low_power_cells</u>	1032
<u>report_low_power_intent</u>	1033
<u>verify_power_structure</u>	1034
<u>write_power_intent</u>	1036
<b>14</b>	
<b><u>Design Manipulation</u></b>	1039
<u>change_link</u>	1041
<u>change_names</u>	1043
<u>clock_gating</u>	1050
<u>delete_unloaded_undriven</u>	1051
<u>edit_netlist</u>	1052
<u>edit_netlist_bitblast_all_ports</u>	1054
<u>edit_netlist_bitblast_port</u>	1055

## Command Reference for Encounter RTL Compiler

---

<u>edit_netlist connect</u>	1056
<u>edit_netlist dedicate_subdesign</u>	1058
<u>edit_netlist delete</u>	1059
<u>edit_netlist disconnect</u>	1060
<u>edit_netlist group</u>	1062
<u>edit_netlist hier_connect</u>	1064
<u>edit_netlist new_design</u>	1065
<u>edit_netlist new_instance</u>	1066
<u>edit_netlist new_port_bus</u>	1068
<u>edit_netlist new_primitive</u>	1069
<u>edit_netlist new_subport_bus</u>	1071
<u>edit_netlist ungroup</u>	1072
<u>edit_netlist uniquify</u>	1073
<u>group</u>	1074
<u>insert_tiehilo_cells</u>	1075
<u>mv</u>	1078
<u>remove_cdn_loop_breaker</u>	1080
<u>reset_design</u>	1082
<u>rm</u>	1083
<u>ungroup</u>	1084
<u>uniquify</u>	1086
<b>15</b>	
<b>Customization</b>	1087
<u>add_command_help</u>	1088
<u>define_attribute</u>	1089
<u>mesg_make</u>	1094
<u>mesg_send</u>	1096
<u>parse_options</u>	1097
<b>A</b>	
<b>Applets</b>	1101
<u>Introduction</u>	1102
<u>applet</u>	1103
<u>applet_avail</u>	1104

## **Command Reference for Encounter RTL Compiler**

---

<u>applet install</u> .....	1106
<u>applet list</u> .....	1107
<u>applet load</u> .....	1108
<u>applet update</u> .....	1109
<u>applet version</u> .....	1110
<u>applet whatis</u> .....	1111
<b><u>Index</u></b> .....	1

# Alphabetical List of Commands

## Symbols

? [71](#)

## A

add\_command\_help [1088](#)  
add\_opcg\_hold\_mux [632](#)  
alias [72](#)  
all\_des\_inps [396](#)  
all\_des\_insts [397](#)  
all\_des\_outs [398](#)  
all\_des\_seqs [399](#)  
all\_lib [401](#)  
all\_lib\_bufs [402](#)  
all\_lib\_ties [403](#)  
all\_connected [394](#)  
all\_inputs [73](#)  
all\_outputs [74](#)  
analyze\_library\_corners [404](#)  
analyze\_scan\_compressibility [633](#)  
analyze\_testability [642](#)  
applet [1103](#)  
applet\_avail [1104](#)  
applet\_install [1106](#)  
applet\_list [1107](#)  
applet\_load [1108](#)  
applet\_update [1109](#)  
applet\_version [1110](#)  
applet\_whatis [1111](#)  
apply\_power\_intent [1016](#)  
apropos [75](#)  
assemble\_design [386](#)  
attribute\_exists [76](#)

## B

basename [38](#)  
build\_rtl\_power\_models [969](#)

## C

cd [39](#)

change\_link [1041](#)  
change\_names [1043](#)  
check\_atpg\_rules [645](#)  
check\_cpf [1019](#)  
check\_design [406](#)  
check\_dft\_pad\_configuration [647](#)  
check\_dft\_rules [648](#)  
check\_floorplan [573](#)  
check\_library [1022](#)  
check\_mbist\_rules [654](#)  
check\_placement [577](#)  
clear [77](#)  
clear\_redline\_terminal [78](#)  
clock\_gating [971](#)  
clock\_gating\_connect\_test [973](#)  
clock\_gating\_declone [974](#)  
clock\_gating\_import [975](#)  
clock\_gating\_insert\_in\_netlist [977](#)  
clock\_gating\_insert\_obs [978](#)  
clock\_gating\_join [980](#)  
clock\_gating\_remove [982](#)  
clock\_gating\_share [984](#)  
clock\_gating\_split [986](#)  
clock\_ports [411](#)  
clock\_uncertainty [314](#)  
commit\_power\_intent [1028](#)  
compare\_sdc [412](#)  
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 [1029](#)  
create\_mode [317](#)  
create\_placement\_blockage [580](#)  
create\_placement\_halo\_blockage [581](#)  
create\_region [582](#)  
create\_routing\_blockage [585](#)  
create\_row [584](#)  
create\_timing\_bin [413](#)  
create\_track [588](#)  
cwd [178](#)  
cwd\_check [179](#)

## Command Reference for Encounter RTL Compiler

cwd create\_check [183](#)  
cwd report\_check [185](#)

### D

date [79](#)  
decrypt [206](#)  
def\_move [589](#)  
define\_attribute [1089](#)  
define\_clock [320](#)  
define\_cost\_group [325](#)  
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\_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 [719, 761](#)  
define\_dft test\_mode [765](#)  
delete\_unloaded\_undriven [1051](#)  
derive\_environment [326](#)  
dft\_trace\_back [769](#)  
dirname [41](#)  
dirs [42](#)  
duplicate\_register [590](#)

### E

edit\_netlist [1052](#)  
edit\_netlist bitblast\_all\_ports [1054](#)  
edit\_netlist bitblast\_port [1055](#)

edit\_netlist connect [1056](#)  
edit\_netlist dedicate\_subdesign [1058](#)  
edit\_netlist disconnect [1060](#)  
edit\_netlist group [1062, 1074](#)  
edit\_netlist hier\_connect [1064](#)  
edit\_netlist new\_design [1065](#)  
edit\_netlist new\_instance [1066](#)  
edit\_netlist new\_port\_bus [1068](#)  
edit\_netlist new\_primitive [1069](#)  
edit\_netlist new\_subport\_bus [1071](#)  
edit\_netlist ungroup [1072](#)  
edit\_netlist uniquify [1073](#)  
elaborate [362](#)  
enable\_transparent\_latches [80](#)  
encrypt [207](#)  
exec\_embedded\_script [81](#)  
exit [83](#)  
export\_critical\_endpoints [210](#)  
external\_delay [328](#)

### F

fanin [415](#)  
fanout [419](#)  
filter [43](#)  
find [46](#)  
fix\_dft\_violations [771](#)  
fix\_scan\_path\_inversions [775](#)

### G

generate\_constraints [332](#)  
generate\_ilm [387](#)  
generate\_ple\_model [592](#)  
generate\_reports [594](#)  
get\_attribute [84](#)  
get\_liberty\_attribute [86](#)  
get\_read\_files [87](#)  
get\_remove\_assign\_options [365](#)  
group [1074](#)  
gui\_balloon\_info [127](#)  
gui\_hide [127](#)  
gui\_hv\_clear [133](#)  
gui\_hv\_get\_file [133](#)  
gui\_hv\_load\_file [133](#)  
gui\_hv\_set\_indicators [134](#)  
gui\_info [127](#)  
gui\_legend [128](#)  
gui\_pv\_airline\_add [142](#)

gui\_pv\_airline\_add\_custom [144](#)  
gui\_pv\_airline\_delete [145](#)  
gui\_pv\_airline\_display [146](#)  
gui\_pv\_airline\_raw\_add [147](#)  
gui\_pv\_airline\_raw\_add\_custom [148](#)  
gui\_pv\_align\_instance\_to\_boundary [149](#)  
gui\_pv\_align\_instances [150](#)  
gui\_pv\_clear [150](#)  
gui\_pv\_connectivity\_airlines [151](#)  
gui\_pv\_deselect [152](#)  
gui\_pv\_display\_collection [153](#)  
gui\_pv\_draw\_box [154](#)  
gui\_pv\_draw\_circle [155, 157](#)  
gui\_pv\_draw\_line [156](#)  
gui\_pv\_get\_design [158](#)  
gui\_pv\_highlight [159](#)  
gui\_pv\_highlight\_hier\_instance [161](#)  
gui\_pv\_highlight\_update [162](#)  
gui\_pv\_label [163](#)  
gui\_pv\_lineup\_instances [164](#)  
gui\_pv\_new\_viewer [164](#)  
gui\_pv\_preferences [165](#)  
gui\_pv\_redraw [166](#)  
gui\_pv\_select [166](#)  
gui\_pv\_selection [167](#)  
gui\_pv\_snapshot [168](#)  
gui\_pv\_steiner\_tree [170](#)  
gui\_pv\_toolbar\_button [171](#)  
gui\_pv\_update [172](#)  
gui\_pv\_zoom\_box [173](#)  
gui\_pv\_zoom\_fit [173](#)  
gui\_pv\_zoom\_in [174](#)  
gui\_pv\_zoom\_out [174](#)  
gui\_pv\_zoom\_to [175](#)  
gui\_raise [129](#)  
gui\_reset [129](#)  
gui\_resume [130](#)  
gui\_selection [130](#)  
gui\_show [130](#)  
gui\_status [131](#)  
gui\_suspend [131](#)  
gui\_sv\_clear [136](#)  
gui\_sv\_cone [136](#)  
gui\_sv\_get\_instance [136](#)  
gui\_sv\_grey [136](#)  
gui\_sv\_highlight [137](#)  
gui\_sv\_load [138](#)  
gui\_sv\_snapshot [139](#)  
gui\_sv\_toolbar\_button [139](#)  
gui\_update [131](#)

## H

hdl\_create [187](#)  
hdl\_create\_binding [188](#)  
hdl\_create\_component [190](#)  
hdl\_create\_implementation [192](#)  
hdl\_create\_library [194](#)  
hdl\_create\_operator [195](#)  
hdl\_create\_package [196](#)  
hdl\_create\_parameter [198](#)  
hdl\_create\_pin [200](#)  
help [88](#)

## I

identify\_domain\_crossing\_pins\_for\_cgic\_and\_scan\_abstracts [776](#)  
identify\_multibit\_cell\_abstract\_scan\_segments [777](#)  
identify\_shift\_register\_scan\_segments [779](#)  
identify\_test\_mode\_registers [781](#)  
include [89](#)  
inout\_mate [51](#)  
insert\_dft [784](#)  
insert\_dft\_boundary\_scan [787](#)  
insert\_dft\_compression\_logic [791](#)  
insert\_dft\_jtag\_macro [806](#)  
insert\_dft\_lockup\_element [810](#)  
insert\_dft\_logic\_bist [811](#)  
insert\_dft\_mbist [817](#)  
insert\_dft\_opcg [823](#)  
insert\_dft\_pmbist [825](#)  
insert\_dft\_ptam [830](#)  
insert\_dft\_rrfa\_test\_points [833](#)  
insert\_dft\_scan\_power\_gating [840](#)  
insert\_dft\_shadow\_logic [843](#)  
insert\_dft\_shift\_register\_test\_points [848](#)  
insert\_dft\_test\_point [849](#)  
insert\_dft\_user\_test\_point [855](#)  
insert\_dft\_wrapper\_cell [857](#)  
insert\_dft\_wrapper\_instruction\_register [863](#)  
insert\_dft\_wrapper\_mode\_decode\_block [865](#)  
insert\_test\_compression [867](#)  
insert\_tiehi0\_cells [1075](#)

## L

lcd [90](#)  
license [91](#)  
license checkin [92](#)  
license checkout [93](#)  
license feature [95](#)  
license list [96](#)  
license version [97](#)  
ll [52](#)  
lls [98](#)  
lpopd [99](#)  
lpushd [100](#)  
lpwd [101](#)  
ls [52](#), [53](#)

## M

man [102](#)  
map\_mbist\_cgc\_to\_cgic [870](#)  
merge\_to\_multibit\_cells [366](#)  
mesg\_make [1094](#)  
mesg\_send [1096](#)  
modify\_power\_domain\_attr [596](#)  
more [103](#)  
move\_blockage [598](#)  
move\_instance [599](#)  
move\_port [600](#)  
move\_region [601](#)  
multi\_cycle [334](#)  
mv [1078](#)

## P

parse\_options [1097](#)  
path\_adjust [338](#)  
path\_delay [342](#)  
path\_disable [345](#)  
path\_group [348](#)  
popd [58](#)  
propagate\_constraints [351](#)  
pushd [60](#)  
pwd [61](#)

## Q

quit [105](#)

## R

read\_db [212](#)  
read\_def [602](#)  
read\_dfm [214](#)  
read\_dft\_abstract\_model [871](#)  
read\_encounter [606](#)  
read\_hdl [218](#)  
read\_ilm [389](#)  
read\_io\_speclist [874](#)  
read\_memory\_view [875](#)  
read\_netlist [224](#)  
read\_pmbist\_interface\_files [877](#)  
read\_power\_intent [1030](#)  
read\_saif [988](#)  
read\_sdc [229](#)  
read\_sdp\_file [607](#)  
read\_spf [608](#)  
read\_tcf [993](#)  
read\_vcd [998](#)  
redirect [106](#)  
remap\_to\_dedicated\_clock\_library [367](#)  
remove\_assigns\_without\_optimization [36](#)  
8  
remove\_cdn\_loop\_breaker [1080](#)  
remove\_clock\_reconvergence [371](#)  
remove\_inserted\_sync\_enable\_logic [372](#)  
replace\_opcg\_scan [878](#)  
replace\_scan [880](#)  
report [422](#)  
report area [427](#)  
report boundary\_opto [429](#)  
report case\_analysis [431](#)  
report cdnLoop\_breaker [432](#)  
report cell\_delay\_calculation [433](#)  
report clock\_gating [434](#)  
report clocks [440](#)  
report congestion [442](#)  
report datapath [443](#)  
report design\_rules [448](#)  
report dft\_chains [449](#)  
report dft\_clock\_domain\_info [454](#)  
report dft\_core\_wrapper [455](#)  
report dft\_registers [459](#)  
report dft\_setup [463](#)  
report dft\_violations [467](#)  
report disabled\_transparent\_latches [470](#)  
report gates [471](#)  
report hierarchy [475](#)  
report instance [477](#)

## Command Reference for Encounter RTL Compiler

---

report low\_power\_cells [480](#)  
report memory [487](#)  
report memory\_cells [488](#)  
report messages [489](#)  
report multibit\_inferencing [492](#)  
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 [483](#)  
reset\_attribute [108](#)  
reset\_design [1082](#)  
reset\_opcg\_equivalent [890](#)  
reset\_scan\_equivalent [891](#)  
resize\_blockage [611](#)  
resize\_region [612](#)  
restore\_congestion\_map [613](#)  
restore\_design [234](#)  
resume [109](#)  
retime [373](#)  
rm [1059, 1083](#)

## S

save\_congestion\_map [614](#)  
sdc\_shell [110](#)  
set\_attribute [111](#)  
set\_compatible\_test\_clocks [892](#)  
set\_opcg\_equivalent [894](#)  
set\_remove\_assign\_options [375](#)  
set\_scan\_equivalent [896](#)  
sh [113](#)  
shell [114](#)  
specify\_cell\_pad [615](#)  
specify\_floorplan [616](#)

specify\_paths [353](#)  
split\_db [236](#)  
state\_retention [1005](#)  
state\_retention  
    connect\_power\_gating\_pins [1006](#)  
state\_retention\_swap [1007](#)  
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 [115](#)  
string\_representation [116](#)  
summary\_table [618](#)  
suppress\_messages [117](#)  
suspend [118](#)  
synthesize [379](#)

## T

test\_super\_thread\_servers [122](#)  
timestat [568](#)

## U

ungroup [1084](#)  
uniquify [1086](#)  
unsuppress\_messages [123](#)  
update\_congestion\_map [620](#)  
update\_gcell\_congestion [621](#)  
update\_gcell\_pin\_density [622](#)  
update\_gcell\_utilization [623](#)  
update\_scan\_chains [898](#)

## V

validate\_constraints [359](#)  
validate\_timing [569](#)  
vdir\_lsearch [62](#)  
verify\_power\_structure [1034](#)  
vname [64](#)

### W

what\_is [65](#)  
what\_is\_list [66](#)  
write\_atpg [900](#)  
write\_bsdl [903](#)  
write\_compression\_macro [906](#)  
write\_db [240](#)  
write\_def [624](#)  
write\_design [243](#)  
write\_dft\_abstract\_model [914](#)  
write\_dft\_rtl\_model [917](#)  
write\_do\_ccd [247](#)  
write\_do\_ccd\_cdc [248](#)  
write\_do\_ccd\_compare\_sdc [249](#)  
write\_do\_ccd\_generate [251](#)  
write\_do\_ccd\_validate [255](#)  
write\_do\_clp [256](#)  
write\_do\_lec [259](#)  
write\_do\_verify\_cdc [263](#)  
write\_encounter [265](#)  
write\_et\_atpg [918, 919, 920, 944](#)  
write\_et\_bsv [926](#)  
write\_et\_dfa [930](#)  
write\_et\_lbist [934](#)  
write\_et\_mbist [938](#)  
write\_et\_no\_tp\_file [943](#)  
write\_ets [275](#)  
write\_ett [276](#)  
write\_forward\_saif [1008](#)  
write\_hdl [278](#)  
write\_io\_speclist [948](#)  
write\_ldb [283](#)  
write\_logic\_bist\_macro [950](#)  
write\_mbist\_testbench [954](#)  
write\_pmbist\_interface\_files [958](#)  
write\_pmbist\_testbench [960](#)  
write\_power\_intent [1036](#)  
write\_saif [1010](#)  
write\_scandef [964](#)  
write\_script [291](#)  
write\_sdc [294](#)  
write\_sdf [298](#)  
write\_sdprj\_file [625](#)  
write\_set\_load [303](#)  
write\_spf [626](#)  
write\_sv\_wrapper [305](#)  
write\_tcf [1012](#)  
write\_template [309](#)

# 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<sup>TM</sup> RTL Compiler. This manual describes each command available within the RTL Compiler shell with their command options.

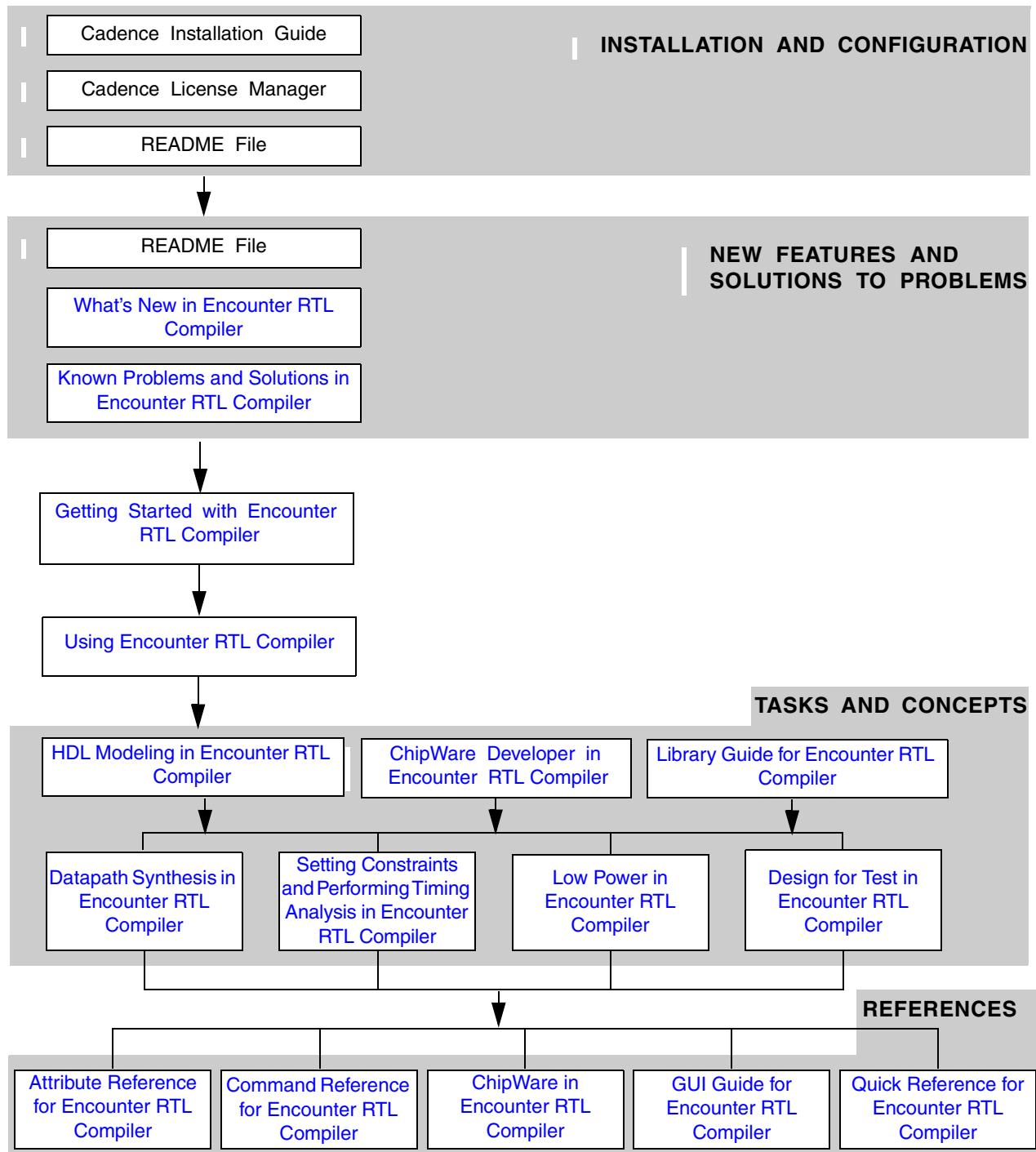
## Additional References

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

- TclTutor, a computer aided instruction package for learning the Tcl language: <http://www.msen.com/~clif/TclTutor.html>.
- TCL Reference, *Tcl and the Tk Toolkit*, John K. Ousterhout, Addison-Wesley Publishing Company
- 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)
- IEEE Standard VHDL Language Reference Manual (IEEE Std. 1076-1987)
- IEEE Standard VHDL Language Reference Manual (IEEE Std. 1076-1993)

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

## How to Use the Documentation Set



## **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.
- **Education services**—Offers instructor-led classes, self-paced Internet, and virtual classroom.
- **University software program support**—Provides you with the latest information to answer your technical questions.

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 PCR or email the message ID you would like improved to:

[rc\\_msg\\_improvement@cadence.com](mailto:rc_msg_improvement@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 our 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. 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 * * -h
```

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 describes the syntax conventions used for the RTL Compiler text commands.

<code>literal</code>	Nonitalic words indicate keywords that you must type literally. These keywords represent command, attribute or option names
<code>arguments and options</code>	Words in italics indicate user-defined arguments or options for which you must substitute a name or a value.
<code> </code>	Vertical bars (OR-bars) separate possible choices for a single argument.
<code>[ ]</code>	Brackets denote options. When used with OR-bars, they enclose a list of choices from which you can choose one.
<code>{ }</code>	Braces denote arguments and are used to indicate that a choice is required from the list of arguments separated by OR-bars. You must choose one from the list  <code>{ argument1   argument2   argument3 }</code>
<code>{ }</code>	Braces in bold-face type must be entered literally.
	Braces, used in Tcl command examples, indicate that the braces must be typed in.
<code>...</code>	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 ( <code>argument...</code> ), you must specify at least one argument, but can specify more.
<code>#</code>	The pound sign precedes comments in command files.

---

# 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 60
- [pwd](#) on page 61
- [vdir\\_lsearch](#) on page 62
- [vname](#) on page 64
- [what\\_is](#) on page 65
- [what\\_is\\_list](#) on page 66

## **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**

<i>pathname</i>	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
      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
rc:/> basename [dirname [dirname $libcell]]
```

### **Related Information**

Related commands: [dirname on page 41](#)

## Command Reference for Encounter RTL Compiler Navigation

cd

cd [ *directory* ]

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

<i>directory</i>	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

## Command Reference for Encounter RTL Compiler

### Navigation

---

#### Related Information

Affects these commands:

[dirs](#) on page 42  
[ls](#) on page 53  
[popd](#) on page 58  
[pushd](#) on page 60  
[pwd](#) on page 61

## **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**

<i>pathname</i>	Specifies the path name of the object, including the object name.
-times <i>integer</i>	Specifies the number of times to apply dirname. <i>Default:</i> 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

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 60

[pwd](#) on page 61

## **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**

<i>attribute_name</i>	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 ( <code>{ }</code> ).
<i>attribute_value</i>	Specifies the value of an attribute to use as filter.
<code>-expr string</code>	Specifies an expression that can be used to compare attribute values. Applies to numerical expressions only.
<code>-invert</code>	Filters out objects that match the expression and returns those that do not.
<i>object_list</i>	Specifies a Tcl list of objects to filter.
<code>-regexp</code>	Overrides the default Tcl glob pattern matching with Tcl regular expression matching.
<code>-special</code>	Use this option when the attribute value contains special characters, such as square brackets.
<code>-vname</code>	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:

## Command Reference for Encounter RTL Compiler

### Navigation

---

```
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 *]]
/libraries/penny/libcells/ANTENNA/
/libraries/penny/libcells/FILL1/
/libraries/penny/libcells/RF1R1WX2/
/libraries/penny/libcells/RF2R1WX2/
```

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

```
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]}{/designs/test/instances_
seq/out_reg[1]}
```

```
rc:/> filter -special dft_test_clock \
{/designs/test/dft/test_clock_domains/clk[0]/clk[0]} $instances
```

## Command Reference for Encounter RTL Compiler

### Navigation

---

```
{/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.

```
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.

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

### Related Information

Affects these commands:

[ls](#) on page 53

[get\\_attribute](#) on page 84

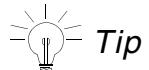
[set\\_attribute](#) on page 111

## 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`.



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`.

## Command Reference for Encounter RTL Compiler

### Navigation

---

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

## Examples

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

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

## Command Reference for Encounter RTL Compiler

### Navigation

---

- 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_  
seq7aluout_reg_0 /designs/alu/instances_seq7aluout_reg_3  
/designs/alu/instances_seq/aluout_reg_5 /designs/alu/instances_seq/aluout_  
reg_2 /designs/alu/instances_seq/aluout_reg_1 /designs/alu/instances_  
seq7zero_reg
```

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

```
rc:/> find des*/alu -port ports_in/*  
{/designs/alu/ports_in/opcode[2]} {/designs/alu/ports_in/opcode[1]}  
{/designs/alu/ports_in/opcode[0]} {/designs/alu/ports_in/data[7]}  
{/designs/alu/ports_in/data[6]} {/designs/alu/ports_in/data[5]}  
{/designs/alu/ports_in/data[4]} {/designs/alu/ports_in/data[3]}  
{/designs/alu/ports_in/data[2]} {/designs/alu/ports_in/data[1]}  
{/designs/alu/ports_in/data[0]} {/designs/alu/ports_in/accum[7]}  
{/designs/alu/ports_in/accum[6]} {/designs/alu/ports_in/accum[5]}  
{/designs/alu/ports_in/accum[4]} {/designs/alu/ports_in/accum[3]}  
{/designs/alu/ports_in/accum[2]} {/designs/alu/ports_in/accum[1]}  
{/designs/alu/ports_in/accum[0]} /designs/alu/ports_in/clock  
/designs/alu/ports_in/ena /designs/alu/ports_in/reset
```

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

```
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]}
```

## Command Reference for Encounter RTL Compiler

### Navigation

---

```
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 support 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+) -messsages * *
```

- 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 (">>").

## Command Reference for Encounter RTL Compiler

### Navigation

---

- 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_3
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_2
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_1
/designs/DTMF_CHIP/instances_hier/DMA_INST/instances_seq/a_reg_0
rc:/designs> echo $rc
rc:/designs>
```

### Related Information

Related command: [clock\\_ports on page 411](#)

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`

## Command Reference for Encounter RTL Compiler

### Navigation

---

#### II

```
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.

## Command Reference for Encounter RTL Compiler

### Navigation

---

## ls

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

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

### 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.
<i>file</i>	Specifies the name of the file to which to list the information.
-long	Lists the contents (long listing) of the directory.
<i>object</i>	Specifies the directory for which you want to list information. <i>Default:</i> current directory
-regexp <i>string</i>	Specifies a regular expression to filter attribute names.
-R	Recursively lists the encountered subdirectories.
-width <i>integer</i>	Specifies the width of the screen that can be used to show the information. <i>Default:</i> 80

### Examples

- The following command lists the contents of the libraries directory:

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

## Command Reference for Encounter RTL Compiler

### Navigation

---

- 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
```

## Command Reference for Encounter RTL Compiler

### Navigation

---

```
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
./          (libcell)
A/          (libpin)
B/          (libpin)
Z/          (libpin)

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

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

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

## Command Reference for Encounter RTL Compiler

### Navigation

---

- 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_Conervative
rc:/designs> get_attribute wireload /designs/async_set_reset_flop_n/
/libraries/slow/wireload_models/sartre18_Conervative
```

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
./          (root)
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
designs/
flows/
hdl_libraries/
libraries/
messages/
mmmc_designs_spec/
object_types7
```

## **Command Reference for Encounter RTL Compiler**

### Navigation

---

#### **Related Information**

Related command: [get\\_attribute](#) on page 84

## **popd**

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 60

## **Command Reference for Encounter RTL Compiler**

### Navigation

---

pwd on page 61

## **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*'.  
pushd: push current dir onto stack and cd to new dir  
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 61

## **pwd**

pwd

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

### **Examples**

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

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

### **Related Information**

Related commands:	<a href="#"><u>dirs</u> on page 42</a>
	<a href="#"><u>ls</u> on page 53</a>
	<a href="#"><u>popd</u> on page 58</a>
	<a href="#"><u>pushd</u> on page 60</a>

## **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**

<i>list</i>	Specifies a list of objects. The objects can be regular objects or can be of type <i>vdir</i> .
<i>object</i>	Specifies the object for which you want to know the position in the list. The object must be of type <i>vdir</i> .  <b>Note:</b> If the object is not of type <i>vdir</i> , 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
```

## Command Reference for Encounter RTL Compiler

### Navigation

---

```
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_lsearch $list $pin
4
```

## Command Reference for Encounter RTL Compiler

### Navigation

---

#### 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**

<i>object</i>	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 65](#)

## **Command Reference for Encounter RTL Compiler**

### **Navigation**

---

## **Command Reference for Encounter RTL Compiler**

### **Navigation**

---

---

## **General**

---

- [?](#) on page 71
- [alias](#) on page 72
- [all\\_inputs](#) on page 73
- [all\\_outputs](#) on page 74
- [apropos](#) on page 75
- [attribute\\_exists](#) on page 76
- [clear](#) on page 77
- [clear\\_redline\\_terminal](#) on page 78
- [date](#) on page 79
- [enable\\_transparent\\_latches](#) on page 80
- [exec\\_embedded\\_script](#) on page 81
- [exit](#) on page 83
- [get\\_attribute](#) on page 84
- [get\\_liberty\\_attribute](#) on page 86
- [get\\_read\\_files](#) on page 87
- [help](#) on page 88
- [include](#) on page 89
- [lcd](#) on page 90
- [license](#) on page 91
- [license\\_checkin](#) on page 92
- [license\\_checkout](#) on page 93

## Command Reference for Encounter RTL Compiler

### General

---

- [license feature](#) on page 95
- [license list](#) on page 96
- [license version](#) on page 97
- [lls](#) on page 98
- [lpopd](#) on page 99
- [lpushd](#) on page 100
- [lpwd](#) on page 101
- [man](#) on page 102
- [more](#) on page 103
- [quit](#) on page 105
- [redirect](#) on page 106
- [reset\\_attribute](#) on page 108
- [resume](#) on page 109
- [sdc\\_shell](#) on page 110
- [set\\_attribute](#) on page 111
- [sh](#) on page 113
- [shell](#) on page 114
- [stop\\_suspend](#) on page 115
- [string\\_representation](#) on page 116
- [suppress\\_messages](#) on page 117
- [suspend](#) on page 118
- [tcl\\_load](#) on page 119
- [test\\_super\\_thread\\_servers](#) on page 122
- [unsuppress\\_messages](#) on page 123

# Command Reference for Encounter RTL Compiler

## General

---

?

? [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

```
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**

<i>alias_name</i>	Specifies the alias name.
<i>command_name</i>	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.

```
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_neg/ports_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[7]}{/designs/top/ports_out/out1[6]}
{/designs/top/ports_out/out1[5]}{/designs/top/ports_out/out1[4]}
{/designs/top/ports_out/out1[3]}{/designs/top/ports_out/out1[2]}
{/designs/top/ports_out/out1[1]}{/designs/top/ports_out/out1[0]}
{/designs/top/ports_out/out2[7]}{/designs/top/ports_out/out2[6]}
{/designs/top/ports_out/out2[5]}{/designs/top/ports_out/out2[4]}
{/designs/top/ports_out/out2[3]}{/designs/top/ports_out/out2[2]}
{/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

<code>-skip_help</code>	Do not include help text in the search results.
<code>-skip_command</code>	Do not include commands in the search results.
<code>-skip_attributes</code>	Do not include attributes in the search results.
<code>string</code>	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 if the specified attribute exists.

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

## **Options and Arguments**

<i>attribute</i>	Specifies the attribute name to be checked.
<i>-path object_path</i>	Specifies the path to the object for which the attribute must be checked.
<i>-type object_type</i>	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
```

## **Command Reference for Encounter RTL Compiler**

### General

---

#### **clear**

clear

Clears the terminal screen.

**clear\_redline\_terminal**

clear\_redline\_terminal

Clears the terminal screen.

## **date**

date

Returns the date 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 470

## **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**

<code>-design <i>string</i></code>	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.
<code>-subdesign <i>string</i></code>	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.

## Command Reference for Encounter RTL Compiler

### General

---

#### Related Information

Related command: [elaborate](#) on page 362

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.



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**

<i>attribute_name</i>	Specifies the name of an attribute whose value you want to retrieve.
<i>object</i>	Specifies the path to the object for which the attribute value should be retrieved.  <i>Default:</i> current working directory
<i>type</i>	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.  <b>Note:</b> 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
```

## Command Reference for Encounter RTL Compiler

### General

---

- 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_Conervative
```

- Using the **get\_attribute wireload** command:

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

returns the following wireload model:

```
/libraries/slow/wireload_models/sartre18_Conervative
```

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 111

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**

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

### **Example**

```
rc:/> get_read_files
{source .7simple_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 .7simple_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

<code>file</code>	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:	<a href="#"><u>hdl_search_path</u></a> <a href="#"><u>lib_search_path</u></a> <a href="#"><u>script_search_path</u></a>
-------------------------------	---

## **lcd**

`lcd directory`

Changes the UNIX working directory to the specified directory.

### **Options and Arguments**

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

### **Examples**

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

`rc:>/ lcd rtl_lab01`

### **Related Information**

Affects these commands:

[ls](#) on page 98  
[lpwd](#) on page 101  
[shell](#) on page 114

## **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:	<a href="#">license checkin</a> on page 92
	<a href="#">license checkout</a> on page 93
	<a href="#">license feature</a> on page 95
	<a href="#">license list</a> on page 96
	<a href="#">license version</a> on page 97

## **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

<code>license</code>	Specifies the license to be checked out. The licenses that can be checked out are: <ul style="list-style-type: none"><li>■ RC_NG100</li><li>■ RTL_Compiler_Adv_Phys_Option</li><li>■ RTL_Compiler_CPU_Accel_Option</li><li>■ RTL_Compiler_Low_Power_Option</li><li>■ RTL_Compiler_Ultra</li><li>■ RTL_Compiler_Ultra_II_Option</li></ul>
<code>-version float</code>	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.
<code>-wait</code>	Waits until a license becomes available. The wait timeout is controlled by the value specified for the <code>-wait</code> option when you started RTL Compiler. The default wait timeout is 600s.  <b>Note:</b> This option does not require the <code>-queue</code> 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
```

## **Command Reference for Encounter RTL Compiler**

### **General**

---

- 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

<i>license</i>	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
<i>-version float</i>	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.

```
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`

IIS

`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 90

Related commands: [lpwd](#) on page 101

[shell](#) on page 114

## **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:	<a href="#">lls on page 98</a>
	<a href="#">lpushd on page 100</a>
	<a href="#">lpwd on page 101</a>

## **lpushd**

`lpushd directory`

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

### **Options and Arguments**

*directory*

Specifies the name of the UNIX directory to be set as target directory.

You can use wildcards when they do not produce an ambiguous reference (more than one match).

### **Example**

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

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

### **Related Information**

Affects these commands:

[lls on page 98](#)

[lpopd on page 99](#)

[lpwd on page 101](#)

## **lpwd**

`lpwd`

Returns the UNIX working directory.

### **Examples**

- The following example shows the current UNIX directory.

```
rc:/> lpwd  
/usr3/verilog_labs
```

### **Related Information**

Affected by this command:      [lcd on page 90](#)

Related commands:      [lls on page 98](#)  
[shell on page 114](#)

#### **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_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] [<design>]...
```

##### 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 ;
NAMECASESENSITIVE 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
```

## Command Reference for Encounter RTL Compiler

### General

---

```
#      + 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**

`quit`

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



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

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

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 ( <code>stdout</code> ) .
-variable	Redirects the output to a Tcl variable.
<i>command</i>	Specifies the command or Tcl code to execute as a quoted string.
<i>target</i>	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"
```

## Command Reference for Encounter RTL Compiler

### General

---

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

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

- The following examples output a timing and a scan\_power gzip compressed report file:

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

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

## **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**

<i>attribute_name</i>	Specifies the attribute that should be returned to its default value. The "*" wildcard character is supported.
<i>object</i>	Specifies the object for which the attribute values should be returned to the default values.
<i>-quiet</i>	Suppresses those messages that indicate which attributes and objects are being affected.
<i>type</i>	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**

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 115  
[suspend](#) on page 118

## **sdc\_shell**

`sdc_shell`

Opens 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**

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

```
rc:/> 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**

<i>attribute_name</i>	Specifies the name of the attribute whose value you want to set.
<i>attribute_value</i>	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.
<i>-lock</i>	Locks the specified attribute's value so that it cannot be changed. The attribute becomes read-only.
<i>objects</i>	Specifies the path(s) to the objects.  <i>Default:</i> current directory
<i>-quiet</i>	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.

<i>type</i>	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 84

## sh

`sh command_string`

Executes a UNIX shell command from the RTL Compiler shell.

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



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

<code>command_string</code>	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`.



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**

<code>command_string</code>	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:	<a href="#">lcd</a> on page 90
Related commands:	<a href="#">lls</a> on page 98 <a href="#">lpwd</a> on page 101

## **stop\_suspend**

`stop_suspend`

Terminates 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 109

[suspend](#) on page 118

## **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**

<i>message_id</i>	Specifies the message identification.
<i>-n integer</i>	Prints the specified message only <i>n</i> number of times. The default value is 0.
<i>-quiet</i>	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/rchap/reg/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/rchap/reg/rc/dft/SMG/native/analysis/test2/LIB
```

### **Related Information**

Related command: [unsuppress\\_messages](#) on page 123

## **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 109](#)  
[stop\\_suspend on page 115](#)

## **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**

<i>filename</i>	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.
<i>packageName</i>	Is the name of the package, and is used to compute the name of an initialization procedure.
<i>interp</i>	Is the path name of the interpreter into which to load the package (see the <code>interp</code> manual entry for details).  If this argument is omitted, it defaults to the interpreter in which the <code>load</code> ( <code>tcl_load</code> ) 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.

## Command Reference for Encounter RTL Compiler

### General

---

The initialization procedure must match the following prototype:

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

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

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

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

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

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

## Bugs

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

### Example

The following is a minimal extension:

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

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

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

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

### See Also

[info sharedlibextension](#), [Tcl\\_StaticPackage\(3\)](#), [safe\(n\)](#)

### Keywords

binary code, loading, safe interpreter, shared library

## **test\_super\_thread\_servers**

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 117](#)

## **Command Reference for Encounter RTL Compiler**

### **General**

---

---

## **GUI Text**

---

- [General GUI Text Commands](#) on page 126
- [HDL Viewer GUI Text Commands](#) on page 132
- [Schematic Viewer GUI Text Commands](#) on page 135
- [Physical Viewer GUI Text Commands](#) on page 140

## **General GUI Text Commands**

- [gui\\_balloon\\_info](#) on page 127
- [gui\\_hide](#) on page 127
- [gui\\_info](#) on page 127
- [gui\\_legend](#) on page 128
- [gui\\_raise](#) on page 129
- [gui\\_reset](#) on page 129
- [gui\\_resume](#) on page 130
- [gui\\_selection](#) on page 130
- [gui\\_show](#) on page 130
- [gui\\_status](#) on page 131
- [gui\\_suspend](#) on page 131
- [gui\\_update](#) on page 131

## **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 129

[gui\\_show](#) on page 130

## **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 titlename [-cancel string] string
```

Creates a GUI legend. This is useful if you need a simple legend to color code information.

**Note:** You can use this command with the Physical Viewer and Schematic Viewer.

### **Options and Arguments**

<i>-cancel string</i>	Specifies a Tcl proc that can be run when the Close button is pressed.
<i>string</i>	Specifies the data set to display in the legend.
<i>-title titlename</i>	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.

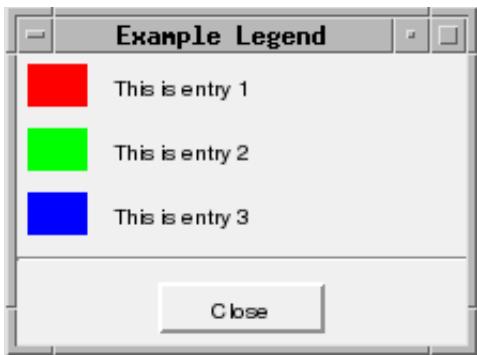
```
proc foo {} {
    puts "Legend cancel"
}
```

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\_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 127](#)  
    [gui\\_show on page 130](#)

## **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**

`gui_resume`

Resumes automatic GUI updates.

### **Related Information**

Related command: [gui\\_suspend](#) on page 131

## **gui\_selection**

`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**

`gui_show [-nosync]`

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 127  
[gui\\_raise](#) on page 129

## **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 130

## **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 130  
[gui\\_suspend](#) on page 131

## HDL Viewer GUI Text Commands

- [gui\\_hv\\_clear](#) on page 133
- [gui\\_hv\\_get\\_file](#) on page 133
- [gui\\_hv\\_load\\_file](#) on page 133
- [gui\\_hv\\_set\\_indicators](#) on page 134

### **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.

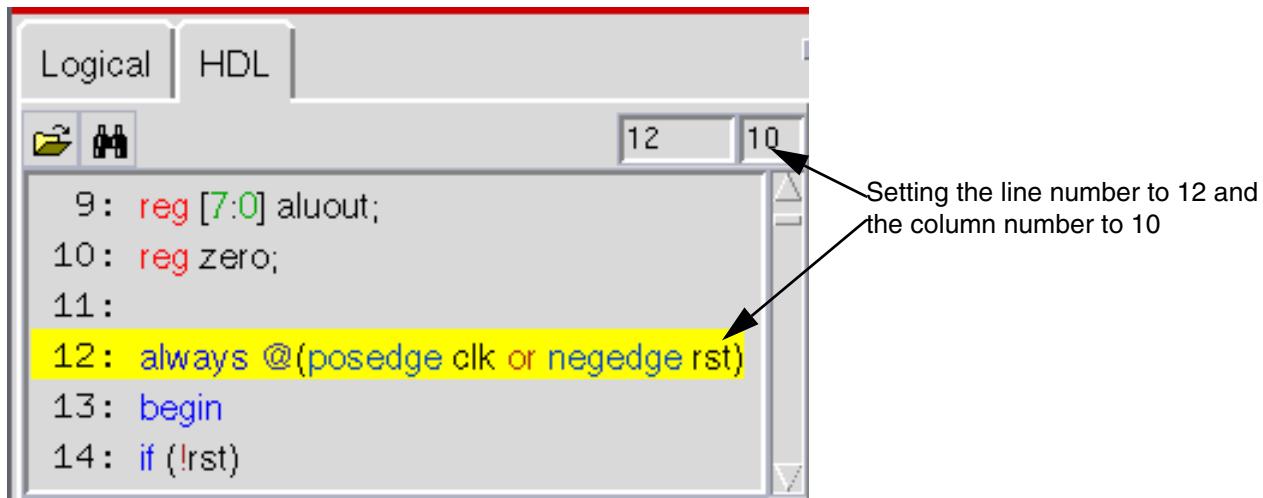
### **Options and Arguments**

<i>-clear</i>	Removes all tag highlighting. If this option is not used, then the specified line number is highlighted.
<i>column_number</i>	Specifies the column number. If the <i>column_number</i> argument is not specified, then the default is 0.
<i>line_number</i>	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 136
- [gui\\_sv\\_cone](#) on page 136
- [gui\\_sv\\_get\\_instance](#) on page 136
- [gui\\_sv\\_grey](#) on page 136
- [gui\\_sv\\_highlight](#) on page 137
- [gui\\_sv\\_load](#) on page 138
- [gui\\_sv\\_snapshot](#) on page 139
- [gui\\_sv\\_toolbar\\_button](#) on page 139

## **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**

<code>instance</code>	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**

<code>-on</code>	Turns on grey mode in the Schematic Viewer.
<code>-off</code>	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**

<code>-append</code>	Appends an object to the highlighted list.
<code>-center</code>	Centers an object in the Schematic Viewer.
<code>-color color</code>	Specifies the color for highlighting an object.
<code>-from {port pin}</code>	Specifies the start point of the highlighted net.
<code>-to {port pin}</code>	Specifies the end point of the highlighted net.
<code>-zoomto</code>	Zooms into the specified port, pin, instance or net.

### **Examples**

- The following Tcl procedure highlights inverters in the Schematic Viewer

```
# highlight inverters
proc highlight_inverters {idir type odir} {
    # only proceed if no object under cursor
    if {$type != "none"} return
    # clear any highlight and selection
    gui_sv_clear
    _find_inverters $idir
}

proc _find_inverters {idir} {
    # search for inverters
    foreach inst [find $idir -maxdepth 2 -instance *comb/*] {
        if {[get_attribute inverter $inst] == "true"} {
            gui_sv_highlight $inst -append -color green
        }
    }
}
```

## Command Reference for Encounter RTL Compiler GUI Text

---

- The following example highlights the falling clock edge flip-flops

```
# highlight falling clock edge flip-flops
proc highlight_falling_edge_ff {idir type odir} {
    # only proceed if no object under cursor
    if {$type != "none"} return
    # clear any highlight and selection
    gui_sv_clear
    _find_falling_edge_ff $idir
}

proc _find_falling_edge_ff {idir} {
    # search for sequential gates with falling clock edge
    foreach inst [find $idir -maxdepth 2 -instance *seq/*] {
        set libcell [get_attribute libcell $inst]
        if {$libcell == ""} continue
        if {[get_attribute flop $libcell] == "true"} {
            foreach libpin [find $libcell -libpin *] {
                if {[get_attribute output $libpin] == "true"} {
                    foreach arc [get_attribute incoming_timing_arcs $libpin] {
                        set liberty [get_attribute liberty_attributes $arc]
                        if {[lsearch $liberty falling_edge] != -1} {
                            gui_sv_highlight $inst -append
                        }
                    }
                }
            }
        }
    }
}
```

### **gui\_sv\_load**

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

Specifies an hierarchical instance or 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**

<i>file</i>	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**

<i>-icon string</i>	Specifies the GIF or PNM file that contains the drawing of the icon.  The icon size is 24x24 and has a gray94 background.
<i>-proc string</i>	Specifies the name of the Tcl procedure to execute.  The procedure must be loaded before you press the button to call it.
<i>-tip string</i>	Specifies the tool tip to be displayed.

## **Physical Viewer GUI Text Commands**

- [gui\\_pv\\_airline\\_add](#) on page 142
- [gui\\_pv\\_airline\\_add\\_custom](#) on page 144
- [gui\\_pv\\_airline\\_delete](#) on page 145
- [gui\\_pv\\_airline\\_display](#) on page 146
- [gui\\_pv\\_airline\\_raw\\_add](#) on page 147
- [gui\\_pv\\_airline\\_raw\\_add\\_custom](#) on page 148
- [gui\\_pv\\_align\\_instance\\_to\\_boundary](#) on page 149
- [gui\\_pv\\_align\\_instances](#) on page 150
- [gui\\_pv\\_clear](#) on page 150
- [gui\\_pv\\_connectivity\\_airlines](#) on page 151
- [gui\\_pv\\_deselect](#) on page 152
- [gui\\_pv\\_display\\_collection](#) on page 153
- [gui\\_pv\\_draw\\_box](#) on page 154
- [gui\\_pv\\_draw\\_circle](#) on page 155
- [gui\\_pv\\_draw\\_line](#) on page 156
- [gui\\_pv\\_draw\\_triangle](#) on page 157
- [gui\\_pv\\_get\\_design](#) on page 158
- [gui\\_pv\\_highlight](#) on page 159
- [gui\\_pv\\_highlight\\_hier\\_instances](#) on page 161
- [gui\\_pv\\_highlight\\_update](#) on page 162
- [gui\\_pv\\_label](#) on page 163
- [gui\\_pv\\_lineup\\_instances](#) on page 164
- [gui\\_pv\\_new\\_viewer](#) on page 164
- [gui\\_pv\\_preferences](#) on page 165
- [gui\\_pv\\_redraw](#) on page 166
- [gui\\_pv\\_select](#) on page 166

## **Command Reference for Encounter RTL Compiler**

### **GUI Text**

---

- [gui\\_pv\\_selection](#) on page 167
- [gui\\_pv\\_snapshot](#) on page 168
- [gui\\_pv\\_steiner\\_tree](#) on page 170
- [gui\\_pv\\_toolbar\\_button](#) on page 171
- [gui\\_pv\\_update](#) on page 172
- [gui\\_pv\\_zoom\\_box](#) on page 173
- [gui\\_pv\\_zoom\\_fit](#) on page 173
- [gui\\_pv\\_zoom\\_in](#) on page 174
- [gui\\_pv\\_zoom\\_out](#) on page 174
- [gui\\_pv\\_zoom\\_to](#) on page 175

## **gui\_pv\_airline\_add**

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

Adds the specified airline to the Physical Viewer.

### **Options and Arguments**

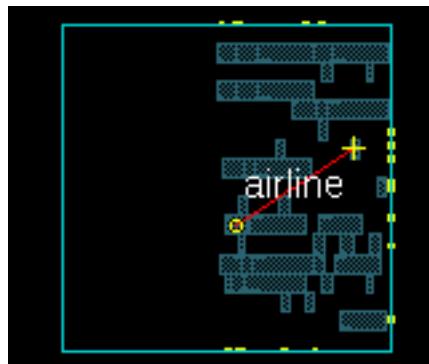
<code>-color color</code>	Specifies the airline color.  <i>Default:</i> blue
<code>-from {port instance}</code>	Specifies the from object.
<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>-label label</code>	Specifies a label to place at the center of the airline.
<code>-name name</code>	Specifies the airline name.
<code>-nodisplay</code>	Specifies not to display the airline.
<code>-to {port instance}</code>	Draws an airline from the center of the from object to the center of the to object.

### **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 164

## **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 <i>color</i>	Specifies the airline color.  <i>Default:</i> blue
-from { <i>instance</i>   <i>port</i> }	Specifies the from object.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
-label <i>label</i>	Specifies a label to place at the center of the airline.
-name <i>name</i>	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 { <i>instance</i>   <i>port</i> }	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 <i>integer</i>	Specifies the airline width.

## Related Information

Related command: [gui\\_pv\\_new\\_viewer](#) on page 164

### **gui\_pv\_airline\_delete**

`gui_pv_airline_delete name [-id integer]`

Deletes airlines from the Physical Viewer with the specified *name*.

#### Options and Arguments

<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>name</code>	Specifies the airline name to be deleted.  The airline name must have been added using the <a href="#">gui_pv_airline_add</a> 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 164

## **gui\_pv\_airline\_display**

`gui_pv_airline_display name [-id integer]`

Displays the specified airline name in the Physical Viewer.

### **Options and Arguments**

<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>name</code>	Specifies the airline name to be displayed.  The airline name must have been added using the <a href="#"><u>gui_pv_airline_add</u></a> 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 164

## **gui\_pv\_airline\_raw\_add**

```
gui_pv_airline_raw_add [-name name] [-id integer]
[-color color] [-label label]
[-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**

<code>-color color</code>	Specifies the airline color.  <i>Default:</i> blue
<code>-fx float</code>	Specifies the x coordinate of the from object.
<code>-fy float</code>	Specifies the y coordinate of the from object.
<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>-label label</code>	Specifies a label to place at the center of the airline.
<code>-name name</code>	Specifies the airline name.
<code>-nodisplay</code>	Specifies not to display the airline.
<code>-tx float</code>	Specifies the x coordinate of the to object.
<code>-ty float</code>	Specifies the y coordinate of the to object.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer](#) on page 164

## **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**

<code>-color color</code>	Specifies the airline color.  <i>Default:</i> blue
<code>-fx float</code>	Specifies the X coordinate of the from object.
<code>-fy float</code>	Specifies the Y coordinate of the from object.
<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>-label label</code>	Specifies a label to place at the center of the airline.
<code>-name name</code>	Specifies the airline name.
<code>-nodisplay</code>	Specifies not to display the airline at the time you create it.
<code>-noglyph</code>	Specifies not to display the airline glyphs.
<code>-nolabel</code>	Specifies not to display the airline label.
<code>-nomiddle</code>	Specifies not to display the airline arrow in the middle.
<code>-tx float</code>	Specifies the X coordinate of the to object.
<code>-ty float</code>	Specifies the Y coordinate of the to object.
<code>-type {0   1   2}</code>	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
<code>-width integer</code>	Specifies the airline width.

## Related Information

Related command: [gui\\_pv\\_new\\_viewer](#) on page 164

### **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.
<i>instance</i>	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.
<i>instance</i>	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 <i>integer</i>	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 164

## **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 <i>integer</i>	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 164

## **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	blockage	bump
cluster	def_pin	gcell
instance	pcell	pdomain
pin	pinstance	port
region	row	

## **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 164

## **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 154

[gui\\_pv\\_draw\\_circle](#) on page 155

[gui\\_pv\\_draw\\_line](#) on page 156

[gui\\_pv\\_draw\\_triangle](#) on page 157

[gui\\_pv\\_highlight](#) on page 159

[gui\\_pv\\_new\\_viewer](#) on page 164

## **gui\_pv\_draw\_box**

```
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	Appends the object to highlight.
-collect	Collects objects to highlight.
-color <i>color</i>	Specifies the highlight color. <i>Default:</i> red
-dbu	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
-label <i>label</i>	Specifies a label to place at the center of the box.
-group <i>name</i>	Displays only those objects in the specified group. These objects were previously marked for display through the <code>gui_pv_highlight -collect</code> command.
-height <i>float</i>	Specifies the height of the box in micrometer.
-llx <i>float</i>	Specifies the x coordinate of the origin of the box in micrometer.
-lly <i>float</i>	Specifies the y coordinate of the origin of the box in micrometer.
-label <i>string</i>	Specifies the object label.
-stipple	Specifies to use stipple fill pattern to highlight the object.
-width <i>float</i>	Specifies the width of the box micrometer.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer](#) on page 164

## **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**

<code>-append</code>	Appends the object to highlight.
<code>-collect</code>	Collects objects to highlight.
<code>-color color</code>	Specifies the highlight color. <i>Default:</i> red
<code>-cx float</code>	Specifies the x coordinate of the center of the circle in micrometer.
<code>-cy float</code>	Specifies the y coordinate of the center of the circle in micrometer.
<code>-dbu</code>	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>-label label</code>	Specifies a label to place at the center of the box.
<code>-group name</code>	Displays only those objects in the specified group. These objects were previously marked for display through the <code>gui_pv_highlight -collect</code> command.
<code>-label string</code>	Specifies the object label.
<code>-radius float</code>	Specifies the radius of the circle in micrometer.
<code>-stipple</code>	Specifies to use stipple fill pattern to highlight the objects.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer on page 164](#)

## **gui\_pv\_draw\_line**

```
gui_pv_draw_circle -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 <i>color</i>	Specifies the highlight color. <i>Default:</i> red
-dbu	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
-fx <i>float</i>	Specifies the x coordinate of the first (or from) point in micrometer.
-fy <i>float</i>	Specifies the y coordinate of the first (or from) point in micrometer.
-group <i>name</i>	Displays only those objects in the specified group. These objects were previously marked for display through the <code>gui_pv_highlight -collect</code> command.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
-label <i>label</i>	Specifies a label to place at the center of the line.
-stipple	Specifies to use stipple fill pattern to highlight the objects.
-tx <i>float</i>	Specifies the x coordinate of the second (or to) point in micrometer.
-ty <i>float</i>	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 164

## **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 <i>color</i>	Specifies the highlight color. <i>Default:</i> red
-dbu	Specifies the coordinates and dimensions in DB units, where one DB unit is 1/1000 micrometer.
-group <i>name</i>	Displays only those objects in the specified group. These objects were previously marked for display through the <i>gui_pv_highlight -collect</i> command.
-height <i>float</i>	Specifies the height of the triangle in micrometer.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
-label <i>string</i>	Specifies a label to place at the center of the triangle.
-llx <i>float</i>	Specifies the x coordinate of the lower left point in micrometer.
-lly <i>float</i>	Specifies the y coordinate of the lower left point in micrometer.
-stipple	Specifies to use stipple fill pattern to highlight the objects.
-width <i>float</i>	Specifies the width of the triangle in micrometer.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer on page 164](#)

## **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 164

## **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

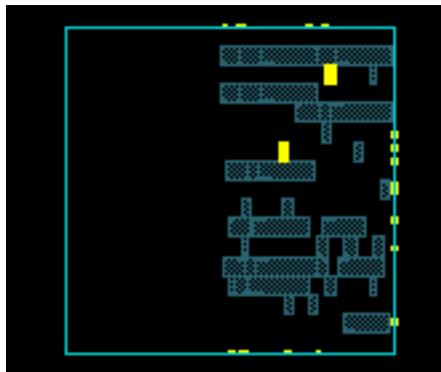
### **Options and Arguments**

-append	Appends the object to highlight.
-collect	Collects objects to highlight
-color <i>color</i>	Specifies the color for highlighting. <i>Default:</i> 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 <i>name</i>	Specifies the group name for the object.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
-label <i>string</i>	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 160):  
rc:/> gui\_pv\_highlight -color yellow g411
- The following command adds the g405 port to the highlighted list (see [Figure 3-4](#) on page 160):  
rc:/> gui\_pv\_highlight -color yellow -append g405

**Figure 3-4 Highlighting an Object in the Physical Viewer**



- The following command specifies the instance `evel` to be highlighted with the `gui_pv_draw_collection` command. The `-group` option indicates that to which group `evel` should belong. In this case, it is `laurence`:

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

## Related Information

Related commands: [gui\\_pv\\_clear](#) on page 150

[gui\\_pv\\_new\\_viewer](#) on page 164

## **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 <i>integer</i>           | Specifies the search depth.   |
| <i>design</i>                   | Specifies the design containing the instances to be highlighted.                          |
| -instances <i>instance_list</i> | Specifies the instances to be highlighted.  |
| -under <i>instance</i>          | 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: [gui\\_pv\\_new\\_viewer](#) on page 164

/// 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**

<code>-color color</code>	Specifies the label color.  <i>Default:</i> white.
<code>-id integer</code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>label</code>	Specifies the label name.
<code>-x float</code>	Specifies the x coordinate for the label.
<code>-y float</code>	Specifies the y coordinate for the label.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer on page 164](#)

## **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.
<i>instance</i>	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 <i>float</i>	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**

<i>design</i>	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**

<code>-color <i>string</i></code>	Specifies the color of the objects of the specified type.
<code>-id <i>integer</i></code>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<code>-name</code>	Specifies to display the name of the objects of the specified type.
<code>-selectable</code>	Specifies that objects of the specified type are selectable.
<code><i>string</i></code>	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_placement, 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.
<code>-style</code>	Specifies the drawing style for the object type. <i>Default:</i> outline.
<code>-visible</code>	Specifies that the objects of the specified type must be visible.

### **Related Information**

Related command: [gui\\_pv\\_new\\_viewer](#) on page 164

## **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 164

## **gui\_pv\_select**

`gui_pv_select object...[-id integer]`

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

You can specify any of the following:

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

### **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 164

## **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 164

## **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.  <b>Note:</b> This option eliminates the need to first turn on the display of the congestion map.
<i>file</i>	Specifies the name of the file in which to save the snapshot.
-id <i>integer</i>	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.  <b>Note:</b> This option eliminates the need to first turn on the display of the utilization map.

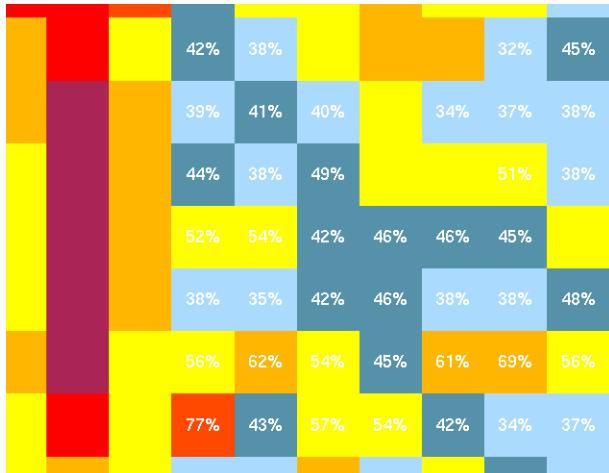
## Command Reference for Encounter RTL Compiler GUI Text

---

### Examples

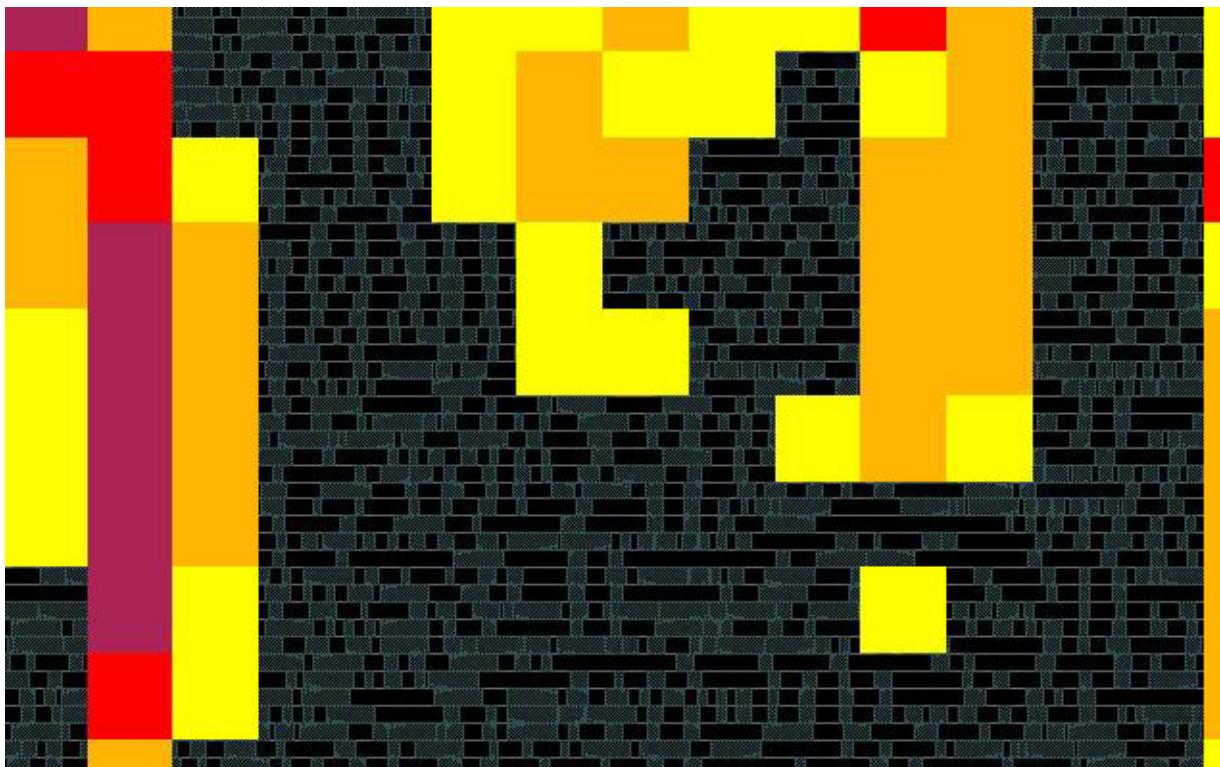
- The following command saves a snapshot of the utilization map in the util.gif file.

```
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 164

### **gui\_pv\_steiner\_tree**

```
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 <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
{ <i>instance</i>   <i>port</i> }	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 164

## **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**

<i>-icon string</i>	Specifies the GIF or PNM file that contains the drawing of the icon.  The icon size is 24x24 and has a gray94 background.
<i>-id integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<i>-proc string</i>	Specifies the name of the Tcl procedure to execute.  The procedure must be loaded before you press the button to call it.
<i>-tip string</i>	Specifies the tool tip to be displayed.

### **Related Information**

Related command:      [gui\\_pv\\_new\\_viewer](#) on page 164

## **gui\_pv\_update**

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

Specifies the objects to be updated in the Physical Viewer.

### **Options and Arguments**

<i>-design design</i>	Specifies the design for which to update the GUI.
<i>-id integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.
<i>string</i>	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 164

## **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 164

## **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 164

## **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 164

## **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 164

## **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 164

## **Command Reference for Encounter RTL Compiler**

### **GUI Text**

---

---

## Chipware Developer

---

- [cwd on page 178](#)
- [cwd check on page 179](#)
- [cwd create check on page 183](#)
- [cwd report check on page 185](#)
- [hdl create on page 187](#)
- [hdl create binding on page 188](#)
- [hdl create component on page 190](#)
- [hdl create implementation on page 192](#)
- [hdl create library on page 194](#)
- [hdl create operator on page 195](#)
- [hdl create package on page 196](#)
- [hdl create parameter on page 198](#)
- [hdl create pin on page 200](#)

## **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:	<a href="#">cwd check</a> on page 179
	<a href="#">cwd create_check</a> on page 183
	<a href="#">cwd report_check</a> on page 185

## 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*
```

## Command Reference for Encounter RTL Compiler

### Chipware Developer

---

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

<code>-effort string</code>	Specifies the effort level. There are two effort levels: <code>low</code> and <code>medium</code> . The default effort level is <code>low</code> .
	<code>Low</code> — CWD linter runs checking rules that are registered as <code>low</code> effort. That is, it runs those checking rules that do not require reading any HDL code (of synthesis models).
	<code>Medium</code> — CWD linter runs checking rules that are registered as <code>low</code> and <code>medium</code> 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 <code>medium</code> 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 <code>hdl_arch</code> of an <code>hdl_impl</code> and examine ordering of pins and parameters.
<code>file</code>	Specifies the filename to store the output of the command.
<code>hdl_lib   hdl_comp   hdl_pack   hdl_oper   hdl_arch   hdl_impl   hdl_bind   hdl_param   hdl_pin</code>	Specifies the <code>hdl_objects</code> to check.
<code>-quiet</code>	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.
<code>-skip {hdl_lib   hdl_comp   hdl_pack   hdl_oper   hdl_arch   hdl_impl   hdl_bind   hdl_param   hdl_pin}</code>	Specify one or more <code>hdl_objects</code> to skip.

## Command Reference for Encounter RTL Compiler

### Chipware Developer

---

-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_prelab_script_location 147    0
non_builtin_implementation_location 147    0
package_default_location     1    0
package_default_location_filesize 1    0
parameter_formula          472    0
pin_bit_width              1136   0
pin_parameter_in_bit_width 1114   0
-----
```

## Command Reference for Encounter RTL Compiler

### Chipware Developer

---

- 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
      Check_name      Passed Failed
      -----
      parameter_formula    2        0
      -----
```

- The following example will check the CW\_add component, but will skip checking its bindings and implementations:

```
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.

It can check correctness, consistency, or completeness of the CWD registration, including availability of UNIX files referred to by the `location` and `sim_model` attributes.

`medium` — The check can read, load, and parse the HDL code of the synthesis models, but it cannot elaborate or synthesize the HDL code. When exercising such a checking rule, the HDL code is automatically loaded before checking is performed.

<code>file</code>	Specifies the filename to store the output of the command.
<code>-force</code>	Removes the existing checkname and adds the current specification. Alternatively, the same checkname will now correspond to the new definition.
<code>-severity integer</code>	<p>Specifies the severity level of the message produced by this checking rule. The possible values (0 through 10) are the same as those for the <code>information_level</code> attribute.</p> <p><code>-severity 0</code>: It is an error message to violate this rule <code>-severity 1</code>: It is a warning message to violate this rule <code>-severity 2</code>: It is a level 2 info message to violate this rule Severity levels 3 through 10 are all info messages at their respective levels.</p>

## 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 \
    in the synthesis model is consistent with what is defined in the \
    registration script"
    -checklist { {HDL_ARCH_PINS_SCANNED check_arch_pin_order} \
    {HDL_COMP_PINS_SCANNED check_component_pin_order} }
```

## 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**

<code>-checkname string</code>	Specifies, by name, a set of checking rules to report.
<code>-checkpoint string</code>	This switch specifies a set of checkpoints to report.
<code>-max_width string</code>	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.
<code>file</code>	Specifies the filename to store the output of the command.

## Command Reference for Encounter RTL Compiler

### Chipware Developer

## Examples

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

```
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.

Check_name	Checkpoint	Check_proc	Effort	Severity	Description
arch_pin_order	HDL_ARCH_PIN S_SCANNED	::cwd::arch_pin order::cwd_proc ium c	0	med	Check whether the order of pins specified in the synthesis model is consistent with what is defined in the registration script
	HDL_COMP_PIN S_SCANNED	::cwd::componen t_pin_order::ch eck_proc			

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

```
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}}
```

Check_name	Checkpoint	Check_proc	Effort	Severity	Description
operator_b	HDL_OPER_BINDIN indings GS_SCANNED	::cwd::operator bindings::che k_proc	1	low	check that for every hdl_bindings defined for the hdl_operator there is at least one attribute is set to false

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

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

- This reports info about all checkpoints:

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

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

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

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

```
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:	<a href="#">hdl_create binding</a> on page 188 <a href="#">hdl_create component</a> on page 190 <a href="#">hdl_create implementation</a> on page 192 <a href="#">hdl_create library</a> on page 194 <a href="#">hdl_create operator</a> on page 195 <a href="#">hdl_create package</a> on page 196 <a href="#">hdl_create parameter</a> on page 198 <a href="#">hdl_create pin</a> on page 200
-------------------	--

## 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

<i>binding_name</i>	Specifies the name of the binding that will be created.
<i>hdl_comp</i>   <i>bindings</i>	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 \
    /hdl_libraries/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:
- [hdl\\_create component](#) on page 190
  - [hdl\\_create implementation](#) on page 192
  - [hdl\\_create library](#) on page 194
  - [hdl\\_create operator](#) on page 195
  - [hdl\\_create package](#) on page 196
  - [hdl\\_create parameter](#) on page 198
  - [hdl\\_create pin](#) on page 200

## **hdl\_create component**

```
hdl_create component component_name
    [hdl_lib | components]
```

Creates a ChipWare component.

### **Options and Arguments**

*component\_name*      Specifies the name of the component that will be created.

*hdl\_lib* | *components*

Specifies the path name of the library that holds this component.

### **Examples**

- The following examples both create a component named `CW_sweet_div`. However, the first example will save run-time over the second by `cd`'ing into the `components` directory and issuing the command:

```
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:

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

## Command Reference for Encounter RTL Compiler

### Chipware Developer

---

#### Related Information

See the following sections in *ChipWare Developer*

- [CWD Component](#)
- [ChipWare Registration](#)

Related commands:

- [hdl\\_create\\_binding](#) on page 188
- [hdl\\_create\\_implementation](#) on page 192
- [hdl\\_create\\_library](#) on page 194
- [hdl\\_create\\_operator](#) on page 195
- [hdl\\_create\\_package](#) on page 196
- [hdl\\_create\\_parameter](#) on page 198
- [hdl\\_create\\_pin](#) on page 200

## **hdl\_create implementation**

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

Creates an implementation for a ChipWare component. All 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.

[-v1995 | -v2001 | -vhdl87 | -vhdl93]

Specifies the language version for the RTL code.

### **Example**

- Both of the following examples create the `krystal` implementation in VHDL 1993 format for the `CW_sweet_div` component. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command:

```
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:

```
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 188
- [hdl\\_create\\_component](#) on page 190
- [hdl\\_create\\_library](#) on page 194
- [hdl\\_create\\_operator](#) on page 195
- [hdl\\_create\\_package](#) on page 196
- [hdl\\_create\\_parameter](#) on page 198
- [hdl\\_create\\_pin](#) on page 200

## **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:

<a href="#"><u>hdl_create binding</u></a> on page 188
<a href="#"><u>hdl_create component</u></a> on page 190
<a href="#"><u>hdl_create implementation</u></a> on page 192
<a href="#"><u>hdl_create operator</u></a> on page 195
<a href="#"><u>hdl_create package</u></a> on page 196
<a href="#"><u>hdl_create parameter</u></a> on page 198
<a href="#"><u>hdl_create pin</u></a> on page 200

## **hdl\_create operator**

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

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

### **Options and Arguments**

<i>operator_name</i>	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:	<a href="#">hdl_create binding</a> on page 188
	<a href="#">hdl_create component</a> on page 190
	<a href="#">hdl_create implementation</a> on page 192
	<a href="#">hdl_create library</a> on page 194
	<a href="#">hdl_create package</a> on page 196
	<a href="#">hdl_create parameter</a> on page 198
	<a href="#">hdl_create pin</a> on page 200

## **hdl\_create package**

```
hdl_create package pkg_name
    -path path_to_pkg
    [hdl_lib | packages]
```

Registers a VHDL package in the ChipWare Developer framework. Packages that are not registered are deleted after elaboration. However, registered packages are never deleted and their information can be further considered during synthesis as opposed to just during elaboration.

Registered packages are in the same location within 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.vhd
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 \
    /hdl_libraries/ieee/packages/numeric_std
rc:/> ls /hdl_libraries/ieee/packages/
./      numeric_std
```

## Related Information

Related commands:

[hdl\\_create binding](#) on page 188

[hdl\\_create component](#) on page 190

[hdl\\_create implementation](#) on page 192

[hdl\\_create library](#) on page 194

[hdl\\_create operator](#) on page 195

[hdl\\_create parameter](#) on page 198

[hdl\\_create pin](#) on page 200

## **hdl\_create parameter**

```
hdl_create parameter parameter_name
    [-hdl_invisible]
    [hdl_comp | parameters]
```

Creates a parameter for a ChipWare component. The created parameter will be a `hdl_param` object type and located under `../component_name/parameters`. The `default hdl_parameter` attribute value for parameters created with this command will be `true`. However, if the `-hdl_invisible` option is specified, the default value becomes `false`.

### **Options and Arguments**

*hdl\_comp* | *parameters*

Specifies the path name of the component that holds this parameter.

`-hdl_invisible`

Specifies that the parameter cannot be accessed from the HDL. The value of the `hdl_parameter` attribute for this parameter becomes `false` with this option.

*parameter\_name*

Specifies the name of the parameter that will be created.

### **Examples**

- Both of the following examples create the `WIDTH` parameter for the `CW_sweet_div` component. However, the first example will save run-time over the second by `cd`'ing into the component name directory and issuing the command:

```
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 188

[hdl\\_create\\_component](#) on page 190

[hdl\\_create\\_implementation](#) on page 192

[hdl\\_create\\_library](#) on page 194

[hdl\\_create\\_operator](#) on page 195

[hdl\\_create\\_package](#) on page 196

[hdl\\_create\\_pin](#) on page 200

## **hdl\_create pin**

```
hdl_create pin pin_name
  {-input | -output | -inout}
  [pins | hdl_oper | hdl_comp]
```

Creates a pin for either a ChipWare component or a synthetic operator. You must specify a pin direction.

### **Options and Arguments**

<code>-inout</code>	Specifies that the created pin will be an bidirectional pin.
<code>-input</code>	Specifies that the created pin will be an input pin.
<code>-output</code>	Specifies that the created pin will be an output pin.
<code><i>pin_name</i></code>	Specifies the name of the pin that will be created.
<code><i>pins</i>   <i>hdl_oper</i>   <i>hdl_comp</i></code>	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 188
- [hdl\\_create component](#) on page 190
- [hdl\\_create implementation](#) on page 192
- [hdl\\_create library](#) on page 194
- [hdl\\_create operator](#) on page 195
- [hdl\\_create package](#) on page 196
- [hdl\\_create parameter](#) on page 198

**Command Reference for Encounter RTL Compiler**  
Chipware Developer

---

---

## Input and Output

---

- [decrypt](#) on page 206
- [encrypt](#) on page 207
- [export\\_critical\\_endpoints](#) on page 210
- [read\\_db](#) on page 212
- [read\\_def](#) on page 213
- [read\\_dfm](#) on page 214
- [read\\_dft\\_abstract\\_model](#) on page 216
- [read\\_encounter](#) on page 217
- [read\\_hdl](#) on page 218
- [read\\_io\\_speclist](#) on page 222
- [read\\_memory\\_view](#) on page 223
- [read\\_netlist](#) on page 224
- [read\\_pmbist\\_interface\\_files](#) on page 226
- [read\\_power\\_intent](#) on page 227
- [read\\_saif](#) on page 228
- [read\\_sdc](#) on page 229
- [read\\_spf](#) on page 231
- [read\\_tcf](#) on page 232
- [read\\_vcd](#) on page 233
- [restore\\_design](#) on page 234
- [split\\_db](#) on page 236
- [write\\_atpg](#) on page 237

## Command Reference for Encounter RTL Compiler

### Input and Output

---

- [write\\_bsdl](#) on page 238
- [write\\_compression\\_macro](#) on page 239
- [write\\_db](#) on page 240
- [write\\_def](#) on page 242
- [write\\_design](#) on page 243
- [write\\_dft\\_abstract\\_model](#) on page 245
- [write\\_dft\\_rtl\\_model](#)
- [write\\_do\\_ccd](#) on page 247
- [write\\_do\\_ccd\\_cdc](#) on page 248
- [write\\_do\\_ccd\\_compare\\_sdc](#) on page 249
- [write\\_do\\_ccd\\_generate](#) on page 251
- [write\\_do\\_ccd\\_validate](#) on page 255
- [write\\_do\\_clp](#) on page 256
- [write\\_do\\_lec](#) on page 259
- [write\\_do\\_verify\\_cdc](#) on page 263
- [write\\_encounter](#) on page 265
- [write\\_et\\_atpg](#) on page 268
- [write\\_et\\_bsv](#) on page 269
- [write\\_et\\_lbist](#) on page 271
- [write\\_et\\_lbist](#) on page 271
- [write\\_et\\_mbist](#) on page 272
- [write\\_et\\_no\\_tp\\_file](#) on page 273
- [write\\_et\\_rrfa](#) on page 274
- [write\\_ets](#) on page 275
- [write\\_ett](#) on page 276
- [write\\_forward\\_saif](#) on page 277
- [write\\_hdl](#) on page 278
- [write\\_io\\_speclist](#) on page 282

## Command Reference for Encounter RTL Compiler

### Input and Output

---

- [write\\_ldb](#) on page 283
- [write\\_logic\\_bist\\_macro](#) on page 284
- [write\\_mbist\\_testbench](#) on page 285
- [write\\_pmbist\\_interface\\_files](#) on page 286
- [write\\_pmbist\\_testbench](#) on page 287
- [write\\_saif](#) on page 289
- [write\\_scandef](#) on page 290
- [write\\_script](#) on page 291
- [write\\_sdc](#) on page 294
- [write\\_sdf](#) on page 298
- [write\\_set\\_load](#) on page 303
- [write\\_spef](#) on page 304
- [write\\_sv\\_wrapper](#) on page 305
- [write\\_tcf](#) on page 308
- [write\\_template](#) on page 309

## **decrypt**

```
decrypt [-keydb path] file
```

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

### **Options and Arguments**

<i>file</i>	Specifies the name of the Tcl file to decrypted and evaluated.
<code>-keydb <i>path</i></code>	Sets the NCProtect_KEYDB environment variable to the directory containing the public key needed to decrypt the file.  If you omit this option, you need to set the NCProtect_KEYDB environment variable before you run the <code>decrypt</code> command.

### **Example**

The first command encrypts the Tcl file `my_script.g`. The second command decrypts the `my_script_encr.g` file. These commands are normally executed in different RC sessions.

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

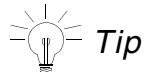
### **Related Information**

Related command:      [encrypt](#) on page 207

#### encrypt

```
encrypt inputfile_name
    [-vlog | -vhdl | -tcl] [-pragma]
    [> file]
```

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



To load encrypted HDL files, use the `read_hdl` command.

The command to source an encrypted Tcl file depends on the file extension of the encrypted Tcl file:

- ❑ If the encrypted file does not have the `.etf` extension, use the `decrypt` command.
- ❑ If the encrypted file has the `.etf` extension, you can use the `source` command.

#### Options and Arguments

<i>input_file_name</i>	Specifies the file to be encrypted.
<i>file</i>	Specifies the name of the encrypted file. By default, the encrypted file is printed to standard out.
-pragma	Specifies to only encrypt the text between the <code>protect begin</code> and <code>protect end</code> NC Protect pragmas.
-tcl	Specifies that the file to be encrypted is a Tcl file. To hide the body of the Tcl scripts, make sure to code procedures using <code>hidden_proc</code> .
-vhdl	Uses VHDL style comments for NC Protect pragmas.
-vlog	Uses Verilog style comments for NC Protect pragmas. This is the default option.

#### Example

- The following example encrypts the `ksable.vhdl` VHDL file, with VHDL constructs, to a file named `ksable_encrypted.vhdl`. The encrypted file is then loaded.

```
rc:/> encrypt -vhdl ksable.vhdl > ksable_encrypted.vhdl
rc:/> read_hdl -vhdl ksable_encrypted.vhdl
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

- The following example illustrates Verilog code with Verilog style NC Protect pragmas. You must specify //pragma protect before specifying the beginning (//pragma protect begin) and ending (//pragma protect end) pragmas.

```
module secret_func (y, a, b);
    parameter w = 4;
    input [w-1:0] a, b;
    output [w-1:0] y;
// pragma protect
// pragma protect begin
    assign y = a & b;
// pragma protect end
endmodule
```

Specifying the -vlog and -pragma options together will only encrypt the text between the pragmas. The following command encrypts the original verilog file (*ori.v*) that contained the NC Protect pragmas. The encrypted file is called *enc.v*.

```
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.

```
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.vhd1*) that contained the NC Protect pragmas. The encrypted file is called *enc.vhd1*:

```
rc:/> encrypt -vhdl -pragma org.vhd1 > enc.vhd1
```

- The following example shows a Tcl script (*test.tcl*) with two procedures: the first procedure starting with *proc*, the second one starting with *hidden\_proc*. When the script is encrypted, no info will be returned for the *im\_hidden* procedure.

```
# pragma protect
# pragma protect begin
proc im_visible {args} {
    # 'info' command will return data for this proc
}
hidden_proc im_hidden {args} {
    # 'info' command will not return data for this proc
}
# pragma protect end
```

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

```
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 206

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **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**

<code>-design <i>string</i></code>	Specifies the module name.
<code>-fe_file <i>string</i></code>	Specifies the First Encounter (FE) slack report that you want to compare.
<code><i>file</i></code>	Specifies the name of the file to write the report.
<code>[-group   -nogroup]</code>	Specifies whether to group endpoints into bins for path_adjust or not. <i>Default:</i> -group
<code>-no_of_bins <i>integer</i></code>	Specifies the number of bins to group the endpoints for compression. <i>Default:</i> 10 bins each for tighten and relax
<code>-percentage_difference <i>integer</i></code>	Specifies the percentage difference between the endpoints to be path adjusted (with the path_adjust command). <i>Default:</i> 70%
<code>-percentage_of_endpoints <i>integer</i></code>	Specifies the percentage of endpoints to be constrained or relaxed. <i>Default:</i> 20%
<code>-rc_file <i>string</i></code>	Specifies the RTL Compiler endpoint report that you want to compare.

## Command Reference for Encounter RTL Compiler Input and Output

**-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**

```
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.

### **Options and Arguments**

<i>db_file</i>	Specifies the name of the database file to be read.
<i>-from_tcl string</i>	Specifies to read the specified Tcl object.
<i>-quiet</i>	Suppresses any warning messages. Error messages are printed.
<i>-verbose</i>	Enables verbose output while reading in the database file.

### **Related Information**

[Using the RTL Compiler Database](#) in *Using Encounter RTL Compiler*

Related commands:

<a href="#">split_db</a> on page 236
<a href="#">write_db</a> on page 240

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

#### **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 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> 0.0000000000000000 </systematic>
    </cell>

    <cell> fflop
      <instance> 0.000000153055338 </instance>
      <systematic> 0.0000000000000000 </systematic>
    </cell>
    <cell> nand2
      <instance> 0.000000044800000 </instance>
      <systematic> 0.0000000000000000 </systematic>
    </cell>
  </cell_probability>
</yield_file>
```

- The following example loads two coefficient files:

```
rc:/> read_dfm test1.dfm
rc:/> read_dfm test2.dfm
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

`-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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

*Default: VHDL-1993*

### 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/  
/hdl_libraries/lib1/architectures:  
./ compute/  
rc:/> ls /hdl_libraries/lib2/architectures/
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

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

#### Related Information

[Reading a Partially Structural Design](#) in *Using Encounter RTL Compiler*.

Affects this command: [elaborate](#) on page 362

Related command: [read\\_netlist](#)

Affects this attribute: [hdl\\_parameter](#)

Affected by these attributes: [hdl\\_search\\_path](#)

[hdl\\_language](#)

[hdl\\_preserve\\_dangling\\_output\\_nets](#)

[hdl\\_verilogDefines](#)

**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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

-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 278

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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### Options and Arguments

<code>file</code>	Specifies the name constraints file to read. You can also specify a file that was compressed with gzip (.gz extension).
<code>-mode mode_name</code>	Reads mode specific constraints for a design.
<code>-no_compress</code>	Turns off advanced compression and compression of exceptions for the remainder of the session.
<code>-stop_on_errors</code>	Stops reading the remainder of the script if an error is encountered during reading of the SDC file.

#### 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.

#### 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:	<a href="#">synthesize</a> on page 379 <a href="#">create_mode</a> on page 317
Related command:	<a href="#">write_sdc</a> on page 294
Affected by these attributes:	<a href="#">break_timing_paths_by_mode</a> <a href="#">enable_break_timing_paths_by_mode</a> <a href="#">detailed_sdc_messages</a>
Related attributes:	<a href="#">embedded_script</a> <a href="#">enable_data_check</a> <a href="#">scale_factor_group_path_weights</a> <a href="#">tim_ignore_data_check_for_non_endpoint_pins</a>

**read\_spef**

Refer to [read\\_spef](#) in [Chapter 10, “Physical.”](#)

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

#### **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**

<code>-db_dir <i>string</i></code>	Specifies the path to the Encounter database directory.
<code>-def <i>file</i></code>	Specifies the path to the DEF file.  If this option is not specified, the tool searches for a <code><i>design</i>.def</code> or <code><i>design</i>.def.gz</code> file in the Encounter database directory. If neither file is found, an error message is issued.
<code>-design_name <i>design</i></code>	Specifies the name of the design.  The design name is the base filename for the output files generated by Encounter tool.
<code>-worst_corner <i>string</i></code>	Specifies the worst case delay corner of all corners defined in the <code>viewdefinitions.tcl</code> file.  This option is required for multi-mode multi-corner designs.  <b>Note:</b> The libraries and capturable 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

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

`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 379](#)

## **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**

<code>-from_tcl file</code>	Specifies to read the specified Tcl object.
<code>-script file</code>	Specifies the name of the script to be written.  If this option is specified without the <code>-to_file</code> 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.
<code>-to_file file</code>	<b>Note:</b> Storing the database in the script takes more memory and results in a larger file.
<code>input_db_file</code>	Specifies the name of the new database 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:

<a href="#">read_db</a> on page 212
<a href="#">write_db</a> on page 240

**write\_atpg**

Refer to [write\\_atpg](#) in Chapter 11, “Design for Test.”

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

#### **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.”](#)

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### 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 1089
- [read\\_db](#) on page 212
- [split\\_db](#) on page 236
- [write\\_design](#) on page 243

**write\_def**

See [write\\_def](#) on page 624 in [Chapter 10, “Physical.”](#)

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **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<sup>®</sup> 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### 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.determinate.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.”

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

#### **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 <i>Generate</i> flow.
propagate	Generates a dofile to create a chip-level SDC file.
validate	Generates a dofile for the <i>Validate</i> flow.

### **Related Information**

[Interfacing with Encounter Conformal Constraint Designer](#) in *Interfacing Between RTL Compiler and Conformal*

Related commands:

- [write\\_do\\_ccd cdc](#) on page 248
- [write\\_do\\_ccd compare\\_sdc](#) on page 249
- [write\\_do\\_ccd generate](#) on page 251
- [write\\_do\\_ccd propagate](#) on page 253
- [write\\_do\\_ccd validate](#) on page 255

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **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**

<i>-design string</i>	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.
<i>file</i>	Specifies the file to which the report must be written.
<i>-logfile file</i>	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
<i>-no_exit</i>	Suppresses the EXIT command at the end of the dofile.
<i>-sdc files</i>	Specifies the SDC files to be read. You can omit this option if you read in the SDC files with the <code>read_sdc</code> command. An error is given when no SDC files are read in either through this command or through <code>read_sdc</code> .

#### **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 <i>string</i>	Specifies the top-level design in RTL Compiler.
-detail	Requests a detailed comparison report.
<i>file</i>	Specifies the file to which the report must be written.
-golden_sdc <i>file</i>	Specifies the UNIX path to the golden SDC file.
-logfile <i>file</i>	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
-netlist <i>file</i>	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 <i>string</i>	Specifies the name of the dofile (script) that must be sourced before the CCD EXIT command.
-pre_read <i>string</i>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
-revised_sdc <i>file</i>	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
```

## **Command Reference for Encounter RTL Compiler**

### Input and Output

---

#### **Related Information**

[Comparing SDC Constraint Files in \*Interfacing Between Encounter RTL Compiler and Encounter Conformal\*](#)

Related command: [compare\\_sdc](#) on page 412

## **write\_do\_ccd generate**

```
write_do_ccd generate
    [-design string] [-netlist string]
    -in_sdc files [-out_sdc file]
    [-slack integer] [-report file]
    [-dfpgen | -fpfgen | -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**

<code>-design <i>string</i></code>	Specifies the top-level design in RTL Compiler.
<code>file</code>	Redirects all the output to the specified file.
<code>-dfpgen</code>	Generates a dofile for the directed false path flow.
<code>-fpfgen</code>	Generates a dofile for the false path flow.
<code>-in_sdc <i>files</i></code>	Specifies the list of SDC files to load.
<code>-logfile <i>file</i></code>	Specifies the name of the CCD logfile.
<code>-mode <i>string</i></code>	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.
<code>-netlist <i>string</i></code>	Specifies the UNIX path to the netlist. This option compares the SDC to the specified netlist instead of the RTL.
<code>-no_exit</code>	Suppresses the <code>exit</code> command at the end of the dofile.
<code>-out_sdc <i>file</i></code>	Specifies the filename to which the identified false paths will be written.  <i>Default:</i> <code>cfp.sdc</code>
<code>-pre_exit <i>string</i></code>	Specifies the name of the dofile (script) that must be sourced before the CCD <code>exit</code> command.
<code>-pre_read <i>string</i></code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

<code>-report file</code>	Specifies the name of the timing report file to be generated. The report will be in CCD format.
<code>-slack integer</code>	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 <code>-report</code> option.
<code>-trv</code>	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**

<b>-block_sdc <i>string</i></b>	Specifies a list of block names with their associated block-level SDC files in the following format:  {{block_name block_sdc_file}...}
<b>-design <i>string</i></b>	Specifies the top-level design in RTL Compiler.
<b><i>file</i></b>	Specifies the file to which the report must be written.
<b>-glue_sdc <i>string</i></b>	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).
<b>-logfile <i>string</i></b>	Creates a separate CCD logfile. You must specify the UNIX path to the file.
<b>-netlist <i>string</i></b>	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
<b>-no_exit</b>	Suppresses the <code>exit</code> command at the end of the dofile.
<b>-out_sdc <i>string</i></b>	Specifies the name of the constraints file that is generated after propagation and integration of the block and glue constraints.  <i>Default:</i> chip.sdc
<b>-partial_chip_sdc <i>string</i></b>	Specifies the name of the partial SDC file that corresponds to the top-level of the design.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

<code>-pre_exit string</code>	Specifies the name of the dofile (script) that must be sourced before the CCD <code>exit</code> command.
<code>-pre_read string</code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
<code>-rule_instance_file string</code>	Specifies the name of the file which defines the rule instances for the Encounter® Conformal® Constraint Designer (CCD) tool.
<code>-rule_instance_template string</code>	<p>Creates a template integration rule instances file.</p> <p>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 <code>-rule_instance_file</code> option.</p>

### 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 ./t1.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**

<code>-design <i>string</i></code>	Specifies the top-level design in RTL Compiler.
<code><i>file</i></code>	Redirects all the output to the specified file.
<code>-init_sequence_file <i>string</i></code>	Specifies the UNIX path to the initialization sequence file for multi-cycle path validation.
<code>-logfile <i>string</i></code>	Specifies the name of the CCD logfile.
<code>-netlist <i>string</i></code>	Specifies the UNIX path to the netlist. This option compares the SDC to the specified netlist instead of the RTL.
<code>-no_exit</code>	Suppresses the <code>exit</code> command at the end of the dofile.
<code>-pre_exit <i>string</i></code>	Specifies the name of the dofile (script) that must be sourced before the CCD <code>exit</code> command.
<code>-pre_read <i>string</i></code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
<code>-sdc <i>string</i></code>	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.

For more information on the Low Power Rule (CLP) Checks, refer to the Encounter® Conformal® Low Power Reference Manual.

### **Options and Arguments**

**-add\_iso\_cell *string***

Specifies the standard cells that CLP should recognize as isolation cells.

**-all**

Specifies to perform all rule checks, whether low-power related or not. This is also the default behavior.

**-clp\_out\_report *string***

Writes the output of the report rule check 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

<code>-ignore_ls_htol</code>	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:  <code>set lowpower option -ignore_high_to_low</code>
<code>-logfile file</code>	Specifies the name of the CLP logfile.
<code>-lp_only</code>	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.
<code>-netlist path</code>	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).
	<i>Default:</i> RTL
<code>-no_exit</code>	Indicates whether to skip the <code>exit</code> command at the end of the dofile. If this option is specified, the following command will be omitted from the dofile:  <code>exit -force</code>
<code>-pre_exit string</code>	Specifies the name of the dofile (script) that must be sourced before the CLP <code>exit</code> command.
<code>-pre_read string</code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
<code>-tclmode</code>	Specifies to generate the dofile as a Tcl script.
<code>-tmp_dir string</code>	Specifies the name of the directory to which the generated files must be written. Its contents will be CLP native commands.
	<i>Default:</i> <code>RC_CLP_design_name_out.do</code>
<code>-verbose</code>	Indicates whether the generated dofile will be verbose. If this option is specified, the <code>report rule check</code> command should write out the intermediate dofile for CLP and then include that file.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### Example

- The following is an example of a CLP dofile:

```
set log file log_file_name -replace
set lowpower option -netlist_style logical

read library -statetable -liberty bn65lplvt_121a/tcbn65lplvtwcl0d90d72.lib \
read design -verilog -sensitive netlist.v

read cpf file cpf_file_name

analyze power domain
rep rule check ISO* LSH* RET* -verbose
exit -force
```

#### Related Information

[Interfacing with Conformal Low Power](#) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

Affected by these attributes:      [wclp\\_lib\\_statetable](#)

## **write\_do\_lec**

```
write_do_lec [-top string]
    [-golden_design string] [-revised_design string]
    [-1801_golden file] [-1801_revised file]
    [-cpf_golden file] [-cpf_revised file]
    [-no_insert_iso_in_dof]
    [-sim_lib string] [-sim_plus_liberty]
    [-cw_sim string]
    [-logfile string] [-env_var string]
    [-pre_read string] [-pre_compare string]
    [-pre_exit string] [-hier | -flat] [-no_exit]
    [-save_session string] [-checkpoint file]
    [-tmp_dir string] [-verbose] [> file]
```

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 <i>file</i>	Specifies the name of the golden (original) 1801 file.  <b>Note:</b> This option only applies to the IEEE-1801 flow.
-1801_revised <i>file</i>	Specifies the name of the revised 1801 file that corresponds to the current state of the design.  <b>Note:</b> This option only applies to the IEEE-1801 flow.
-checkpoint <i>file</i>	Specifies the name of the checkpoint file to generate.
-cpf_golden <i>file</i>	Specifies the name of the golden (original) file.  If omitted, the file will be inferred from the cpf_files design attribute.  <b>Note:</b> This option only applies to the CPF flow.
-cpf_revised <i>file</i>	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_cpf command.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

**Note:** This option only applies to the CPF flow.

<code>-cw_sim string</code>	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:  <code>{V1995 V2001 SV VHDL1987 VHDL1993}</code>
	<i>Default:</i> Language specified in the <code>read_hdl</code> command.
<code>-env_var string</code>	Specifies the names and values of UNIX environment variables to be used in library, design, and log filenames in the dofile.
<code>file</code>	Redirects all the output to the specified file.
<code>-flat</code>	Performs a flattened Conformal LEC comparison.
<code>-golden_design string</code>	Specifies the UNIX path to an alternative golden design.  If the file was loaded into RTL Compiler (using either <code>read_hdl</code> or <code>read_netlist</code> ), the tool knows the language format of the file.  Otherwise, the tool assumes that the format of the file is Verilog-1995.
<code>-hier</code>	Performs a hierarchical Conformal LEC comparison.
<code>-logfile string</code>	Specifies the name of the Conformal LEC logfile.
<code>-no_exit</code>	Does not add the <code>exit</code> command to the end of the dofile.
<code>-no_insert_iso_in_dof</code>	Prevents that the <code>-insert_isolation</code> option is added to the LEC <code>commit power intent</code> 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 <code>commit_power_intent</code> command)
<code>-pre_compare string</code>	Specifies an extra dofile that must be sourced from the dofile before the LEC <code>compare</code> command.
<code>-pre_exit string</code>	Specifies an extra dofile that must be sourced from the dofile before the LEC <code>exit</code> command.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

<code>-pre_read string</code>	Specifies an extra dofile that must be sourced from the dofile before the library and design are read.
<code>-revised_design string</code>	Specifies the UNIX path to the revised design.
<code>-save_session string</code>	Specifies the filename to save the LEC session.
<code>-sim_lib string</code>	Specifies the simulation library in Verilog 1995.
<code>-sim_plus_liberty</code>	Specifies that the simulation library is an addition to the synthesis library.
<code>-tmp_dir string</code>	Specifies the name of the directory to which the generated files must be written.
<code>-top string</code>	Specifies the name of the top-level design in RTL Compiler.
<code>-verbose</code>	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](#) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

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

[More About the write\\_do\\_lec dofile](#) in *Interfacing Between Encounter RTL Compiler and Encounter Conformal*

Related command:

[write\\_power\\_intent](#) on page 1036

## Command Reference for Encounter RTL Compiler

### Input and Output

---

Affected by these attributes:

[boundary optimize invert hier pins](#)  
[wlec add noblack 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 <i>Categorization</i> flow.
-design <i>string</i>	Specifies the name of the top-level design in RTL Compiler.
<i>file</i>	Specifies a specific dofile filename.
-logfile <i>string</i>	Specifies a specific logfile name.
-no_exit	Suppresses the exit command in the dofile.
-sdc <i>string</i>	Specifies a list of the SDC files.
-validate	Specifies to generate a dofile for the <i>Validation</i> 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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### 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**

```
write_encounter design [-basename string] [-tcf]
    [-reference_globals_file globals_file |
     -reference_config_file config_file]
    [-gzip_files] [-ignore_scan_chains] [-ignore_msv]
    [-floorplan string] [-lef lef_files] [design]
```

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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### Options and Arguments

<code>-basename string</code>	Specifies the directory path name and base filename for the output data. The default directory is <code>./rc_enc_des</code> and the default filename without the extension is <code>rc</code> .
<code>design</code>	Specifies a particular design for which to write out information. Only one design can be specified at a time.
<code>-floorplan string</code>	Specifies the extension for the file containing the floorplan. The valid extensions are: <code>.def</code> —DEF <code>.pde</code> —PDEF <code>.fp</code> —Encounter floorplan
<code>-gzip_files</code>	Compresses the netlist and constraints in <code>.gz</code> 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.  <b>Note:</b> Since Global Timing Debug (in Encounter Timing System) does not handle compressed SDC files, the <code>write_encounter</code> command will not compress SDC files.
<code>-ignore_scan_chains</code>	If specified, the scan DEF file will not be written and the scan reorder directives will not be included in the setup file.
<code>-ignore_msv</code>	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.
<code>-lef lef_files</code>	Specifies a particular physical library or libraries to use. The physical libraries will have the <code>.lef</code> extension. The contents of the LEF library that was specified with the <code>lef_library</code> attribute will be used if this option is not specified.
<code>-reference_config_file config_file</code>	Specifies a reference Encounter configuration file to use as a template for the generated configuration file.  <b>Note:</b> The configuration file is used in the Legacy Configuration Data Flow.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

**-reference\_globals\_file *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 317

[read\\_encounter](#) on page 217

**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\_ibist**

Refer to [write\\_et\\_ibist](#) 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.”](#)

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

#### **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**

<code>-default</code>	Generates a simple ETS run script. The script will contain the following ETS commands: <code>read_lib</code> , <code>read_verilog</code> , <code>set_top_module</code> , <code>read_sdc</code> , and <code>report_timing</code> .
<code>file</code>	Redirects the output to the specified file.
<code>-netlist string</code>	Specifies the UNIX path of the file containing the gate-level netlist.
<code>-ocv</code>	Adds an extra <code>set_timing_derate</code> command into the ETS run script. This option can only be specified with the <code>-sdf</code> option.
<code>-post_include string</code>	Specifies the UNIX path of the include file that contains ETS commands that need to be added after the <code>report_timing</code> command in the run file generated by <code>write_ets</code> .
<code>-pre_include string</code>	Specifies the UNIX path of the include file that contains ETS commands that need to be added before the <code>report_timing</code> command in the run file generated by <code>write_ets</code> .
<code>-sdc string</code>	Specifies the UNIX path of the SDC file.
<code>-sdf string</code>	Specifies the UNIX path of the SDF file. If this option is specified, the <code>read_sdf</code> , <code>set_analysis_mode</code> , and <code>set_op_cond</code> commands will be added to the ETS run script.
<code>-spef string</code>	Specifies the UNIX path of the SPEF file. If this option is specified, the <code>read_spef</code> , <code>set_analysis_mode</code> , and <code>set_op_cond</code> 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**

<i>design</i>	Specifies the design for which the constraints must be generated.
<i>file</i>	Specifies the name of the file to which the constraints must be written.
<code>-dc</code>	Writes the constraints that are DC and PT compatible, which means that commands not listed in the SDC specification may be written out.
<code>-ett</code>	Writes an Encounter True Time Clock Constraints file.
<code>-strict</code>	Writes only constraints in SDC format.
<code>-version {1.1 1.3 1.4}</code>	Specifies the SDC version to use to write SDC constraints.

**write\_forward\_saif**

Refer to [write\\_forward\\_saif](#) in Chapter 12, “Low Power Synthesis.”

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **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 <i>logic abstract model</i> .
-depth <i>integer</i>	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.  <i>Default:</i> infinite
{ <i>design</i>   <i>subdesign</i> }	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.
<i>file</i>	Specifies the file to which the output must be written.  <i>Default:</i> 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

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 <i>intermediate</i> netlist with additional information to facilitate formal verification with Encounter® Conformal® Equivalence Checking. See <a href="#">Related Information</a> 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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

- 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:	<a href="#">elaborate</a> on page 362 <a href="#">synthesize</a> on page 379
Affected by these attributes:	<a href="#">optimize_merge_flops</a> <a href="#">optimize_merge_latches</a> <a href="#">optimize_merge_seq</a> <a href="#">write_sv_port_wrapper</a> <a href="#">write_vlog_bit_blast_bus_connections</a> <a href="#">write_vlog_bit_blast_constants</a> <a href="#">write_vlog_bit_blast_mapped_ports</a> <a href="#">write_vlog_bit_blast_tech_cell</a> <a href="#">write_vlog_convert_onebit_vector_to_scalar</a> <a href="#">write_vlog_declare_wires</a> <a href="#">write_vlog_empty_module_for_logic_abstract</a> <a href="#">write_vlog_line_wrap_limit</a> <a href="#">write_vlog_no_negative_index</a> <a href="#">write_vlog_port_association_style</a> <a href="#">write_vlog_preserve_net_name</a> <a href="#">write_vlog_top_module_first</a>

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

write\_vlog\_unconnected\_port\_style  
write\_vlog\_wor\_wand

**write\_io\_speclist**

Refer to [write\\_io\\_speclist](#) in [Chapter 11, “Design for Test.”](#)

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### **write\_ldb**

```
write_ldb  
    library outputfile -opt_level {0|1}
```

Writes out a compiled library.

#### **Options and Arguments**

<i>library</i>	Specifies the name of the library to dumped in ldb format.
-opt_level{0 1}	Specifies whether to dump the original library in gzip format with the ldb file . <ul style="list-style-type: none"><li>■ 0 dumps the source library in zipped form with the compiled library</li><li>■ 1 does not dump the source library</li></ul>
	<i>Default:</i> 0
<i>outputfile</i>	Specifies the name of the file for the dumped library.

**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 [-hdl]
    [-analyze_all_scan_chains [-dont_overlay_segments]]
    [design] [> file]
```

Generates a script that contains the timing for all modes and the design rule constraints of the design. If you used DFT functionality, the script will also contain any test constraints that were applied, as well as any objects that were created in the design as a result of inserting DFT logic such as boundary scan, building the fullscan chains, and inserting scan chain compression logic.

The resulting script can subsequently be used to examine the current design constraints, or it can be read back into 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### 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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

#### Related Information

Affected by these commands:

- [create\\_mode](#) on page 317
- [define\\_clock](#) on page 320
- [define\\_cost\\_group](#) on page 325
- [define\\_dft\\_scan\\_chain](#) on page 739
- [external\\_delay](#) on page 328
- [multi\\_cycle](#) on page 334
- [path\\_adjust](#) on page 338
- [path\\_delay](#) on page 342
- [path\\_disable](#) on page 345
- [path\\_group](#) on page 348

## **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**

<code>design</code>	Specifies the name of the design for which to write the SDC constraints.
<code>file</code>	Specifies the name of the file to which to write the SDC constraints.
<code>-exclude cmd_list</code>	Specifies the commands that must not be written to the SDC file.
<code>-mode mode</code>	Writes out mode specific constraints for a design.
<code>-no_split</code>	Prevents printing the SDC commands over several lines.
<code>-strict</code>	Writes out commands that are specifically listed in the SDC specification. If you do not use this option, the <code>write_sdc</code> command outputs commands that are DC and PT compatible, which means that commands not listed in the SDC specification may be written out. See <a href="#">Examples</a> for the difference in results when using the <code>-strict</code> option.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

```
-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

## 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:

## Command Reference for Encounter RTL Compiler

### Input and Output

```
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 abc1]
rc:/> write_sdc -exclude "set_clock_latency set_clock_uncertainty \
set_case_analysis create_clock"

or

rc:/> write_sdc -exclude {set_clock_latency set_clock_uncertainty \
set_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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

```
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 317  
[read\\_sdc](#) on page 229

# Command Reference for Encounter RTL Compiler

## Input and Output

## **write\_sdf**

```
write_sdf [-version {OVI 3.0 | OVI 2.1}]
           [-precision non_negative_integer]
           [-timescale {ps | ns}] [-delimiter character]
           [-celltiming {all | none | nochecks}]
           [-interconn {port | interconnect [-no_empty_cells]}]
           [-edges {edged | check_edge}] [-condelse]
           [-nonegchecks] [-no_escape] [-nosplit_timing_check]
           [-no_input_port_nets] [-no_output_port_nets]
           [-recrem {merge_always | merge_when_paired | split}]
           [-setup {merge_always | merge_when_paired | split}]
           [-design] [> file]
```

The command generates a Standard Delay Format (SDF) file that analysis and verification tools or timing simulation tools can use for delay annotation. The SDF file specifies the delay of all the cells and interconnects in the design in the Standard Delay Format. Specifically, it includes the delay values for all the timing arcs of a given cell in the design.

**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.

`a11`—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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

`-edges {edged | check_edge}`

Specifies the edges values.

`check_edge`—Keeps edge specifiers on timing check arcs but does not add edge specifiers on combinational arcs.

`edged`—Keeps edge specifiers on timing check arcs as well as combinational arcs.

*Default:* edged

`file`

Specify the SDF file name.

`-interconn {port | interconnect}`

Specifies the construct to use for writing out net delays.

`port`—Writes out the net delays using the PORT construct.

`interconnect`—Writes out the net delays using the INTERCONNECT construct.

*Default:* port

`-no_escape`

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

`-nonegchecks`

Converts all negative timing check values to 0 . 0.

`-no_empty_cells`

Suppresses writing out empty cell descriptions.

`-no_input_port_nets`

Suppresses writing out nets connected to input ports.

`-no_output_port_nets`

Suppresses writing out nets connected to output ports.

`-nosplit_timing_check`

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

`-precision non_negative_integer`

Specifies the number of digits appearing after the decimal point in the output SDF file.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

-recrem [merge\_always | merge\_when\_paired | split]

Determines how RECOVERY and REMOVAL timing checks are written.

**merge\_always**—Always writes combined RECREM checks, even if either of the RECOVERY or REMOVAL check does not exist.

For example, if only a RECOVERY check exists, the tool writes a combined RECREM check in the SDF file as follows:

```
RECREM (value) (value) () ()
```

**split**—Splits the RECREM checks into a RECOVERY check and a REMOVAL check.

**merge\_when\_paired**—Writes combined RECREM checks only when both RECOVERY and REMOVAL checks exist.

For example, if both a RECOVERY and a REMOVAL check exist, the tool writes a combined RECREM check in the SDF file as follows:

```
RECREM (value) (value) (value) (value)
```

However, if only the RECOVERY check exists, the tool only writes the RECOVERY check in the SDF file:

```
RECOVERY (value) (value)
```

*Default:* merge\_always

-setuphold [merge\_always | merge\_when\_paired | split]

Determines how SETUP and HOLD timing checks are written.

**merge\_always**—Always writes combined SETUPHOLD checks, even if either of the SETUP or HOLD check does not exist.

For example, if only a SETUP check exists, the tool writes a combined SETUPHOLD check in the SDF file as follows:

```
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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

For example, if both a **SETUP** and a **HOLD** check exist, the tool writes a combined **SETUPHOLD** check in the SDF file as follows:

```
SETUPHOLD (value) (value) (value) (value)
```

However, if only the **SETUP** check exists, the tool only writes the **SETUP** check in the SDF file:

```
SETUP (value) (value)
```

*Default:* merge\_always

**-timescale {ps | ns}**

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

*Default:* ps

**-version {OVI 3.0 | OVI 2.1}**

Specifies whether to generate SDF version 2.1 or 3.0.

*Default:* OVI 3.0

## Examples

- The following example writes out the SDF file, `areid.sdf`, with the “/” delimiting character:

```
rc:/> synthesize -to_mapped  
rc:/> write_sdf -delimiter "/" > areid.sdf
```

- The following report illustrates two **TIMINGCHECK** examples: the first only writes out the maximum delays while the second writes out all the delays.

```
(TIMINGCHECK  
(HOLD D (posedge CK) (":0.0))  
(SETUP D (posedge CK) (":0.452))  
)  
  
(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))  
)
```

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

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\_spf**

See [write\\_spf](#) 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%<sub>s</sub>* 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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

**-update\_design\_name**

Renames the main design module and uses the value of the **-subdesign\_suffix** option as suffix to the design name.

### Example

Consider the following Verilog RTL input in test.v.

```
// test.v
typedef struct {
    logic x;
    logic y;
} pair;

module test( input pair in1, output out1);
    assign out1 = in1.x & in1.y;
endmodule
```

Assume the Verilog input file is synthesized through the script test.g:

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

```
// test.vc : the file output by RC
module test(\in1[x] , \in1[y] , out1);
    input \in1[x] , \in1[y] ;
    output out1;
    wire \in1[x] , \in1[y] ;
    wire out1;
    wire n_0;
    inv1 g8(.A (n_0), .Y (out1));
    nand2 g9(.A (\in1[x] ), .B (\in1[y] ), .Y (n_0));
endmodule
```

To avoid this port mismatch, the script should be altered as shown below:

```
## test.g
set_attr library tutorial.lib
read_hdl -sv test.v
set_attr hdl_sv_module_wrapper true /
elaborate
synthesize -to_map
write_sv_wrapper -update_design_name -subdesign_suffix _core test > test_wrap.v
write_hdl > test.vc
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

This results in the following output:

```
// File test_wrap.v
typedef struct {
    logic x;
    logic y;
} pair;

module test(
    input pair in1,
    output logic out1);

    test 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;
    invl 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 ( <i>template.cpf</i> ). The template CPF file is read in the template script with the <i>read_cpf</i> 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 <i>-dft</i> and <i>-power</i> 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 <i>define_clock</i> and the <i>external_delay</i> commands.
<i>-outfile string</i>	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.

## Command Reference for Encounter RTL Compiler

### Input and Output

---

-power	Writes out power attributes and commands.
-retime	Writes out retiming attributes and commands.
-simple	Writes out a simple template script.
	You cannot use this option with the <code>-split</code> , <code>-area</code> , <code>-dft</code> , <code>-cpf</code> , <code>-multimode</code> , <code>-physical</code> , <code>-power</code> , <code>-retime</code> , or <code>-full</code> options.
-split	<p>Writes out a template script with a separate setup file which contains the root attributes and setup variables.</p> <p>If you specified the <code>-dft</code> option with the <code>-split</code> option, an additional file is created which contains the DFT design attributes, test clock and scan chain information.</p> <p>If you specified the <code>-power</code> option with the <code>-split</code> option, an additional file is created which contains the leakage and dynamic power, and clock-gating setup information.</p>
-yield	Writes out a template script for yield.

## 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
```

## Command Reference for Encounter RTL Compiler

### Input and Output

---

- 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
```

## **Command Reference for Encounter RTL Compiler**

### **Input and Output**

---

---

# **Constraints**

---

- [clock\\_uncertainty](#) on page 314
- [create\\_mode](#) on page 317
- [define\\_clock](#) on page 320
- [define\\_cost\\_group](#) on page 325
- [derive\\_environment](#) on page 326
- [external\\_delay](#) on page 328
- [generate\\_constraints](#) on page 332
- [multi\\_cycle](#) on page 334
- [path\\_adjust](#) on page 338
- [path\\_delay](#) on page 342
- [path\\_disable](#) on page 345
- [path\\_group](#) on page 348
- [propagate\\_constraints](#)
- [specify\\_paths](#) on page 353
- [validate\\_constraints](#) on page 359

## **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]
```

## Command Reference for Encounter RTL Compiler

### Constraints

---

- 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.

**Note:** You cannot use this command in a CPF flow.

The command returns the directory path to the `mode` object that it creates. You can find the objects created by the `create_mode` command in:

```
/designs/design/modes
```

### **Options and Arguments**

<code>-default</code>	Designates the specified mode as the default mode.  In this case, you can specify only mode with the <code>-name</code> option.
<code>-design design</code>	Specifies the name of the design for which you want to create a mode.
<code>-name mode_list</code>	Specifies the names of the modes to be created. Unless a mode is assigned to be the default mode using the <code>-default</code> 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 "mode1 mode2" -design design_name
```

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

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

## Command Reference for Encounter RTL Compiler

### Constraints

---

- 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:	<a href="#">define_clock</a> on page 320 <a href="#">external_delay</a> on page 328 <a href="#">multi_cycle</a> on page 334 <a href="#">path_adjust</a> on page 338 <a href="#">path_delay</a> on page 342 <a href="#">path_disable</a> on page 345 <a href="#">path_group</a> on page 348 <a href="#">read_sdc</a> on page 229 <a href="#">report_clocks</a> on page 440 <a href="#">report_timing</a> on page 541 <a href="#">specify_paths</a> on page 353 <a href="#">write_sdc</a> on page 294
Related commands:	<a href="#">derive_environment</a> on page 326 <a href="#">report_summary</a> on page 535 <a href="#">write_encounter</a> on page 265 <a href="#">write_script</a> on page 291
Sets this attribute:	<a href="#">default</a>
Related attributes:	(instance) <a href="#">disabled_arcs_by_mode</a> (pin/port) <a href="#">external_delays_by_mode</a> (design-instance) <a href="#">latch_borrow_by_mode</a> (design-instance/pin) <a href="#">latch_max_borrow_by_mode</a>

## **Command Reference for Encounter RTL Compiler**

### Constraints

---

(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**

```
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

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-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)

Defines a clock waveform for a mode.

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.

You can apply clock waveforms to input ports of your design, hierarchical pins, clock pins of sequential cells in your design, a combination, or to no objects at all.

`-rise integer`

Determines together with the `-divide_rise` option the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a fraction of the period and is derived by dividing `-rise` by `-divide_rise`.

*Default:* 0 (rising edge is at the start of the period)

## Examples

- The following example defines a clock for a design with top module `alu`.

```
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:

```
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:

```
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:

```
rc:/> define_clock -period 10000 -name 100MHz -rise 2 -divide_rise 3 \
==> -fall 1 -divide_fall 3
```

**Note:** The `-divide_rise` and `-divide_fall` options allow you to precisely define when the clock transitions occur. In some cases RTL Compiler needs this precise definition to compute the correct timing constraints for paths that are launched by one clock and captured by another.

## Command Reference for Encounter RTL Compiler

### Constraints

---

- 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 411  
[external\\_delay](#) on page 328  
[multi\\_cycle](#) on page 334  
[path\\_adjust](#) on page 338  
[path\\_delay](#) on page 342  
[path\\_disable](#) on page 345  
[path\\_group](#) on page 348  
[report\\_clocks](#) on page 440  
[report\\_qor](#) on page 523

## Command Reference for Encounter RTL Compiler

### Constraints

---

[specify\\_paths](#) on page 353  
[read\\_sdc](#) on page 229  
[report\\_clocks](#) on page 440  
[report\\_summary](#) on page 535  
[report\\_timing](#) on page 541  
[synthesize](#) on page 379  
[write\\_encounter](#) on page 265  
[write\\_script](#) on page 291  
[write\\_sdc](#) on page 294

Related command: [create\\_mode](#) on page 317

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**

<code>-design <i>design</i></code>	Specifies the name of the design for which you want to define the cost group.
<code>-name <i>string</i></code>	Specifies the name of the cost group. <i>Default:</i> grp_x
<code>-weight <i>integer</i></code>	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. <i>Default:</i> 1

### **Examples**

The following example assigns a weight factor of 1 to cost group I20 for the top-level design.

```
rc:/> define_cost_group -name I20 -weight 1
/designs/.../timing/cost_groups/I20
```

### **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:	<a href="#">path_group</a> on page 348 <a href="#">report_timing</a> on page 541 <a href="#">write_script</a> on page 291 <a href="#">write_script</a> on page 291
Sets this attribute:	<a href="#">weight</a>

## derive\_environment

```
derive_environment [-name string] [-sdc_only] 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

<i>instance</i>	Specifies the name of the instance whose environment (constraints) you want to derive.
<code>-name <i>string</i></code>	Specifies the name of the target design.
<code>-sdc_only</code>	<p>Specifies that the tool only apply constraints to the new design that can be expressed in SDC.</p> <p>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.</p> <p>By default, the <code>derive_environment</code> command uses RTL Compiler's more powerful constraint language that produces a more accurate timing view, but is not understood by other tools.</p>

## Command Reference for Encounter RTL Compiler

### Constraints

---

#### 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](#)

- |                           |   |
|---------------------------|---|
| Affected by this command: | <a href="#"><u>path_adjust</u></a> on page 338<br><a href="#"><u>create_mode</u></a> on page 317  |
| Affects these commands:   | <a href="#"><u>report_timing</u></a> on page 541<br><a href="#"><u>synthesize</u></a> on page 379 |
| Sets this attribute:      | <a href="#"><u>precluded_path_adjusts</u></a>   |

## **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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

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.

Constrains ports and pins by mode in a design.

`-mode mode_name`  
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.

{*pin* | *port*}

Specifies delay for timing start points and end points, which can be primary ports, pins of sequential instances, and pins of unresolved references.

Use the [break\\_timing\\_paths](#) attribute to make a non-start/end point a start/end point, which then can be used with the [external\\_delay](#) command.

### Examples

- The following example specifies an input delay of 300 picoseconds on all bits of port *a* relative to the falling edge of clock *clock1*:

```
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 *a* relative to the rising edge (default) of clock *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:      [define\\_clock](#) on page 320

## Command Reference for Encounter RTL Compiler

### Constraints

---

Affects these commands:	<a href="#">report qor</a> on page 523 <a href="#">report timing</a> on page 541 <a href="#">synthesize</a> on page 379 <a href="#">write encounter</a> on page 265 <a href="#">write sdc</a> on page 294 <a href="#">write script</a> on page 291
Related commands:	<a href="#">create mode</a> on page 317 <a href="#">define clock</a> on page 320 <a href="#">derive environment</a> on page 326 <a href="#">multi cycle</a> on page 334 <a href="#">path adjust</a> on page 338 <a href="#">path delay</a> on page 342 <a href="#">path disable</a> on page 345 <a href="#">path group</a> on page 348 <a href="#">specify paths</a> on page 353
Affects these attributes:	<a href="#">External Delay Attributes</a>
Related attributes:	(pin/port) <a href="#">external delays by mode</a>

## **generate\_constraints**

```
generate_constraints [-rtl] [-netlist string]
                     [-slack integer] [-report string]
                     [-in_sdc string] [-out_sdc string]
                     {-fpfgen | -dfpfgen | -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**

<code>-dfpfgen</code>	Generates the false paths from Directed False Path Generation flow.
<code>file</code>	Specifies the name of the file to write the report.
<code>-fpfgen</code>	Generates the false paths from False Path Generation flow.
<code>-in_sdc <i>string</i></code>	Specifies the UNIX path to the SDC files.
<code>-logfile <i>string</i></code>	Creates a separate CCD logfile. You must specify the UNIX path to the file.
<code>-mode <i>string</i></code>	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.
<code>-netlist <i>string</i></code>	Specifies the UNIX path to the netlist.
<code>-out_sdc <i>string</i></code>	Specifies the name for the RTL Compiler generated SDC file. <i>Default:</i> cfp.sdc
<code>-report <i>string</i></code>	Specifies the name of the timing report file to be generated for the design. The file will be in the CCD format.
<code>-rtl</code>	Indicates that the verification should happen against the RTL.

## Command Reference for Encounter RTL Compiler

### Constraints

---

-slack <i>integer</i>	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`:

```
rc:/> generate_constraints -fpfgen -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`:

```
rc:/> generate_constraints -dfpfgen -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 359

## **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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-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`

## Command Reference for Encounter RTL Compiler

### Constraints

---

- 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 353  
[synthesize](#) on page 379  
[write\\_encounter](#) on page 265  
[write\\_sdc](#) on page 294  
[write\\_script](#) on page 291

Related commands:

[create\\_mode](#) on page 317  
[define\\_clock](#) on page 320  
[external\\_delay](#) on page 328  
[path\\_adjust](#) on page 338  
[path\\_delay](#) on page 342  
[path\\_disable](#) on page 345  
[path\\_group](#) on page 348  
[specify\\_paths](#) on page 353

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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

-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 <i>mode_name</i>	Modifies path constraints for a specified mode.
-name <i>string</i>	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 <u>find</u> command) and for recognizing the exception in reports.
	<i>Default:</i> adj_n
-paths <i>string</i>	Specifies the paths to which the exception should be applied. The string argument should be created using the <u>specify_paths</u> command. The -paths option cannot be used in conjunction with any of the other three path selection options (-from, -through, -to).
-through { <i>instance</i>   <i>port</i>   <i>pin</i> }	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 { <i>instance</i>   <i>external_delay</i>   <i>clock</i>   <i>port</i>   <i>pin</i> }	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:

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

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

```
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`:

```
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:	<a href="#">report timing</a> on page 541 <a href="#">report qor</a> on page 523 <a href="#">specify_paths</a> on page 353 <a href="#">synthesize</a> on page 379 <a href="#">write_script</a> on page 291 <a href="#">write_encounter</a> on page 265 <a href="#">write_sdc</a> on page 294 <a href="#">write_script</a> on page 291
Related commands	<a href="#">define_clock</a> on page 320 <a href="#">external_delay</a> on page 328

## **Command Reference for Encounter RTL Compiler**

### Constraints

---

[multi\\_cycle](#) on page 334  
[path\\_delay](#) on page 342  
[path\\_disable](#) on page 345  
[path\\_group](#) on page 348  
[specify\\_paths](#) on page 353

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**

<code>-delay <i>integer</i></code>	Specifies the delay constraint value, in picoseconds, for paths that meet the path selection criteria.
<code>-from {<i>instance   external_delay   clock   port   pin</i>}</code>	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.
<code>-lenient</code>	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.
<code>-mode <i>mode_name</i></code>	Creates a timing exception object for the specified mode that lets you specify the timing constraint for paths that meet the path selection criteria.

## Command Reference for Encounter RTL Compiler

### Constraints

---

<code>-name <i>string</i></code>	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 <a href="#">find command</a> ) and for recognizing the exception in reports.
	<i>Default:</i> <code>del_n</code>
<code>-paths <i>string</i></code>	Specifies the paths to which the exception should be applied. The <i>string</i> argument should be created using the <a href="#">specify_paths command</a> . The <code>-paths</code> option cannot be used in conjunction with any of the other three path selection options ( <code>-from</code> , <code>-through</code> , <code>-to</code> ).
<code>-through {<i>instance</i>   <i>port</i>   <i>pin</i>}</code>	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 <code>-through</code> 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.
<code>-to {<i>instance</i>   <i>external_delay</i>   <i>clock</i>   <i>port</i>   <i>pin</i>}</code>	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
```

## Command Reference for Encounter RTL Compiler

### Constraints

---

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:	<a href="#">report qor</a> on page 523 <a href="#">report timing</a> on page 541 <a href="#">specify_paths</a> on page 353 <a href="#">synthesize</a> on page 379 <a href="#">write_script</a> on page 291 <a href="#">write_sdc</a> on page 294
-------------------------	---

Related commands:	<a href="#">create_mode</a> on page 317 <a href="#">define_clock</a> on page 320 <a href="#">external_delay</a> on page 328 <a href="#">multi_cycle</a> on page 334 <a href="#">path_adjust</a> on page 338 <a href="#">path_disable</a> on page 345 <a href="#">path_group</a> on page 348 <a href="#">specify_paths</a> on page 353
-------------------	--

Sets these attributes:	<a href="#">Exception Attributes</a>
------------------------	--------------------------------------

## **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*

## Command Reference for Encounter RTL Compiler

### Constraints

---

-paths <i>string</i>	Specifies the paths to which the exception should be applied. The string argument should be created using the <code>specify_paths</code> command. The <code>-paths</code> option cannot be used in conjunction with any of the other three path selection options ( <code>-from</code> , <code>-through</code> , <code>-to</code> ).
[ <code>-setup</code>   <code>-hold</code> ]	Limits the exception to either setup timing analysis or hold timing analysis.  By default, the exceptions applies to both.
<code>-through {instance   port   pin}</code>	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 <code>-through</code> 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.
<code>-to {instance   external_delay   clock   port   pin}</code>	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
```

## Command Reference for Encounter RTL Compiler Constraints

---

- 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:	<a href="#"><u>report qor</u></a> on page 523 <a href="#"><u>report timing</u></a> on page 541 <a href="#"><u>specify_paths</u></a> on page 353 <a href="#"><u>synthesize</u></a> on page 379 <a href="#"><u>write_encounter</u></a> on page 265 <a href="#"><u>write_script</u></a> on page 291 <a href="#"><u>write_sdc</u></a> on page 294
Affected by these commands:	<a href="#"><u>define_clock</u></a> on page 320 <a href="#"><u>multi_cycle</u></a> on page 334
Related commands:	<a href="#"><u>create_mode</u></a> on page 317 <a href="#"><u>define_clock</u></a> on page 320 <a href="#"><u>external_delay</u></a> on page 328 <a href="#"><u>path_adjust</u></a> on page 338 <a href="#"><u>path_delay</u></a> on page 342 <a href="#"><u>path_group</u></a> on page 348 <a href="#"><u>specify_paths</u></a> on page 353
Sets these attributes:	<a href="#"><u>Exception Attributes</u></a>

## path\_group

```
path_group
  {-from {instance|external_delay|clock|port|pin}...
   |-through {instance|port|pin}...[-through...]...
   |-to {instance|external_delay|clock|port|pin}...}...
   |-paths string}
  [-group string]
  [-lenient] [-mode mode_name] [-name string]
```

Assigns paths that meet the path selection criteria to a cost group. Paths can be selected using the `-from`, `-through`, `-to`, or `-paths` options. You must provide at least one of these four options. The `-paths` option cannot be used in conjunction with any of the other three path selection options.

### Options and Arguments

<code>-from {instance   external_delay   clock   port   pin}</code>	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.
<code>-group string</code>	Specifies the name of the cost group (defined with <code>define_cost_group</code> ) to which the path is added.
<code>-lenient</code>	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.
<code>-mode mode_name</code>	Assigns paths for a specified mode that meet the path selection criteria to a cost group.
<code>-name string</code>	Associates a name with the specified timing exception.
<code>-paths string</code>	Specifies the paths to which the exception should be applied. The string argument should be created using the <code>specify_paths</code> command. The <code>-paths</code> option cannot be used in conjunction with any of the other three path selection options ( <code>-from</code> , <code>-through</code> , <code>-to</code> ).
<code>-through {instance   port   pin}</code>	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 325  
[report\\_qor](#) on page 523  
[report\\_timing](#) on page 541  
[specify\\_paths](#) on page 353  
[synthesize](#) on page 379  
[write\\_encounter](#) on page 265  
[write\\_script](#) on page 291  
[write\\_sdc](#) on page 294

Related commands:

[create\\_mode](#) on page 317  
[define\\_clock](#) on page 320

## **Command Reference for Encounter RTL Compiler**

### Constraints

---

[external\\_delay](#) on page 328

[multi\\_cycle](#) on page 334

[path\\_adjust](#) on page 338

[path\\_delay](#) on page 342

[path\\_disable](#) on page 345

[specify\\_paths](#) on page 353

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**

<i>-block_sdc string</i>	Specifies a list of block names with their associated block-level SDC files in the following format:  {{block_name <i>block_sdc_file</i> }...}
<b>Note:</b> You need to specify the path to the SDC files. If the paths contain Tcl variables, use the format shown in the <a href="#">Examples</a> on page 352.	
<i>file</i>	Specifies the file to which the report must be written.
<i>-glue_sdc string</i>	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).
<i>-logfile string</i>	Creates a separate CCD logfile. You must specify the UNIX path to the file.
<i>-netlist string</i>	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
<i>-out_sdc string</i>	Specifies the name of the constraints file that is generated after propagation and integration of the block and glue constraints.  <i>Default:</i> chip.sdc
<i>-partial_chip_sdc string</i>	Specifies the name of the partial SDC file that corresponds to the top-level of the design.

## Command Reference for Encounter RTL Compiler

### Constraints

---

**-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 "{block1 $WORKINGDIR/block1.sdc} \
{block2 $WORKINGDIR/block2.sdc} ..."
```

or

```
propagate_constraints -block_sdc [list \
[list block1 ${WORKINGDIR}/block1.sdc] \
[list block2 ${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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

**-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.

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-through_fall_pin {pin | port | instance}`

Specifies a Tcl list of points that a path must traverse. The path is restricted to fall transitions at the indicated points. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-through_rise_pin {pin | port | instance}`

Specifies a Tcl list of points that a path must traverse. The path is restricted to rise transitions at the indicated points. Points to traverse can be ports, hierarchical pins, pins on a sequential/mapped combinational cells, or sequential/mapped combinational instances.

You can repeat the `-through` option to require that a path first must traverse one of the objects in the first set, then pass through one of the objects in the second set, and so on.

`-to {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-to_fall_clock {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to capture at the falling edge of an ideal clock waveform. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

`-to_fall_pin {pin | port | clock | external_delay | instance}`

Specifies a Tcl list of end points for the paths. The paths are restricted to fall transitions at the end points. The end points can be output ports of your design, input pins of flip-flops, clock objects, or a combination of these, instances, or output ports to which specified external delay timing exceptions apply.

## Command Reference for Encounter RTL Compiler

### Constraints

---

`-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]
```

## Command Reference for Encounter RTL Compiler

### Constraints

---

#### 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:	<a href="#"><u>multi_cycle</u></a> on page 334 <a href="#"><u>path_adjust</u></a> on page 338 <a href="#"><u>path_delay</u></a> on page 342 <a href="#"><u>path_disable</u></a> on page 345 <a href="#"><u>path_group</u></a> on page 348 <a href="#"><u>report qor</u></a> on page 523 <a href="#"><u>report timing</u></a> on page 541 <a href="#"><u>write_encounter</u></a> on page 265 <a href="#"><u>write_script</u></a> on page 291 <a href="#"><u>write_sdc</u></a> on page 294
Related commands:	<a href="#"><u>create_mode</u></a> on page 317 <a href="#"><u>define_clock</u></a> on page 320 <a href="#"><u>external_delay</u></a> on page 328
Affects this attribute:	<a href="#"><u>paths</u></a>

## **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**

<code>file</code>	Specifies the name of the file to write the report.
<code>-init_sequence_file string</code>	Specifies the UNIX path to the initialization sequence file for MCP validation.
<code>-logfile string</code>	Creates a separate CCD logfile. You must specify the UNIX path to the file.
<code>-netlist string</code>	Specifies the UNIX path to the netlist. This option also indicates that the validation should happen against the netlist (as opposed to the RTL).
<code>-rtl</code>	Indicates that the validation should happen against the RTL (as opposed to the netlist).
<code>-sdc string</code>	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`

## Command Reference for Encounter RTL Compiler

### Constraints

---

#### Related Information

[Using the Validate Flow without Dofiles in Interfacing between Encounter RTL Compiler and Encounter Conformal](#)

Related command: [generate\\_constraints](#) on page 332

---

## Elaboration and Synthesis

---

- [elaborate](#) on page 362
- [get\\_remove\\_assign\\_options](#) on page 365
- [merge\\_to\\_multibit\\_cells](#) on page 366
- [remap\\_to\\_dedicated\\_clock\\_library](#) on page 367
- [remove\\_assigns\\_without\\_optimization](#) on page 368
- [remove\\_clock\\_reconvergence](#) on page 371
- [remove\\_inserted\\_sync\\_enable\\_logic](#) on page 372
- [retime](#) on page 373
- [set\\_remove\\_assign\\_options](#) on page 375
- [synthesize](#) on page 379

## 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 }
```

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

```
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 218

Affected by this attribute: [library](#)

# Command Reference for Encounter RTL Compiler

## Elaboration and Synthesis

### **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 368

set remove assign options on page 375

# Command Reference for Encounter RTL Compiler

## Elaboration and Synthesis

## **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

<i>design</i>	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**

```
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 <i>clock</i>	Specifies the name of the clocks for which to map the clock-path logic to its library cell collection.
<i>design</i>	Specifies the design for which to perform the remapping.
-dont_add_preserve	<p>Prevents that the instances on the clock path are marked preserved.</p> <p>By default the <code>preserve</code> attribute is set to <code>true</code> for the instances on the clock path.</p>
-effort {low medium high}	<p>Specifies the effort level for remapping and resizing.</p> <ul style="list-style-type: none"><li>■ <code>low</code>—Performs a one-to-one replacement. This works well when the given set of cells is sufficient to replace the complete clock-path logic.</li><li>■ <code>medium</code>—Performs partial one-to-one replacement. Replaces only those instances which have a suitable replacement in the provided library set for the clock.</li><li>■ <code>high</code>—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).</li></ul>

*Default:* medium

## **remove\_assigns\_without\_optimization**

```
remove_assigns_without_optimization
  [-buffer_or_inverter <libcell>]
  [-no_buffers_use_inverters]
  [-allow_unloaded_buffers]
  [-use_inverted_signal]
  [-ignore_preserve_map_size_ok]
  [-ignore_preserve_setting]
  [-include_local_constant_assigns]
  [-dont_respect_boundary_optimization]
  [-preserve_dangling_nets]
  [-dont_skip unconstrained_paths]
  [-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.



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.

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

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.

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

`-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**

<code>-clocks <i>clock</i></code>	Specifies the name of the clocks for which to remove the clock path reconvergence.  By default, the command optimizes the networks for all clocks.
<code><i>design</i></code>	Specifies the design for which to perform the removal of clock reconvergence.
<code>-effort {low high}</code>	 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.  <i>Default:</i> low
<code>-ports <i>port_list</i></code>	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] \\\n[find 7 -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**

<i>file</i>	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:      [synthesize](#) on page 379

## 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:

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

## Related Information

### [Retiming the Design in Using Encounter RTL Compiler](#)

Related command:      [synthesize](#) on page 379

Affected by these attributes:      [dont\\_retime](#)  
[retime](#)  
[retime\\_async\\_reset](#)  
[retime\\_effort\\_level](#)  
[retime\\_hard\\_region](#)  
[retime\\_move\\_mux\\_loop\\_with\\_reg](#)  
[retime\\_optimize\\_reset](#)  
[retiming\\_clocks](#)

Related attributes:      [retime\\_original\\_registers](#)  
[retime\\_reg\\_naming\\_suffix](#)  
[retime\\_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.

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

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.

**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)`.

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

`-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}]
            [-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}**

Specifies the effort to use during optimization.

*Default:* medium

**-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.

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

	<p><b>Note:</b> This option applies only when you specify -auto_identify_shift_register.</p> <p><b>Default:</b> Longest detected shift register segment</p>
-shift_register_min_length <i>integer</i>	<p>Specifies the minimum sequential length of the shift register for auto-identification.</p>
	<p><b>Note:</b> This option applies only when you specify -auto_identify_shift_register.</p> <p><b>Default:</b> 2</p>
-spatial	<p>Performs a quick placement to optimize the mapped gates.</p> <p><b>Note:</b> You must have access to the Encounter® place and route tool to run this command option.</p>
-to_clock_gated	<p>Inserts clock-gating logic in an elaborated design, maps the clock-gating logic and performs RTL optimization on the rest of the design.</p> <p>This option cannot be used with any of the following options: -to_generic, -to_mapped, -to_placed or -spatial.</p> <p><b>Note:</b> For clock-gating logic to be inserted, you must enable the lp_insert_clock_gating root attribute.</p> <p><b>Note:</b> Can only be used when you start from RTL. No clock-gating will be inserted if you start from a netlist.</p>
-to_generic	<p>Performs RTL optimization.</p> <p>This is the default option if the RTL design has not been optimized yet.</p> <p><b>Note:</b> 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.</p>
-to_mapped	<p>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.</p> <p>This is the default option when the design is in the generic or mapped state.</p>

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

**-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**

Current Design State		
RTL	Generic	Mapped
■ RTL Optimization	■ Mapping ■ Incremental Optimizations	■ Unmapping ■ Remapping ■ Incremental Optimizations
(same as -to_generic)	(same as -to_mapped)	(same as -to_mapped)

- The following example limits synthesis to a single design `main`:

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

- The following example maps multiple designs at the same time:

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

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

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

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

---

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**

<b>Current Design State</b>		
<b>RTL</b>	<b>Generic</b>	<b>Mapped</b>
■ RTL Optimization	■ nothing	■ Unmapping

- The following example requests mapping of the design:

```
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**

<b>Current Design State</b>		
<b>RTL</b>	<b>Generic</b>	<b>Mapped</b>
■ RTL Optimization	■ Mapping	■ Unmapping
■ Mapping	■ Incremental Optimizations	■ Remap
■ Incremental Optimizations		■ Incremental Optimizations

- [Table 7-4](#) on page 382 illustrates how the `-incremental` option affects the design:

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

<b>Options</b>	<b>Current Design State</b>	
	<b>RTL/Generic</b>	<b>Mapped</b>
-to_generic -incremental	Disable -incremental and proceed	Unmapping
-to_mapped -incremental	Disable -incremental and proceed	Incremental Optimization
-incremental	Disable -incremental and proceed	Incremental Optimization

## Command Reference for Encounter RTL Compiler

### Elaboration and Synthesis

- 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**

Options	Current Design State		
	RTL	Generic	Mapped
<code>-to_placed</code>	RTL Optimization Mapping Placement Post-placement incremental optimizations	Mapping Placement Post-placement incremental optimizations	Placement Post-placement incremental optimizations
<code>-to_placed</code> <code>-incremental</code>	(same as <code>-to_placed</code> )	(same as <code>-to_placed</code> )	(same as <code>-to_placed</code> )

### Related Information

#### [Inserting Clock Gating in Low Power in Encounter RTL Compiler](#)

Affects these commands:

[report area](#) on page 427

[report clock\\_gating](#) on page 434

[report datapath](#) on page 443

[report design\\_rules](#) on page 448

[report gates](#) on page 471

[report power](#) on page 507

[report summary](#) on page 535

[report timing](#) on page 541

Related Command

[remove\\_assigns\\_without\\_optimization](#) on page 368

Affected by these attributes:

[auto\\_ungroup](#)

[iopt\\_force\\_constant\\_removal](#)

[iopt\\_sequential\\_resynthesis](#)

## **Command Reference for Encounter RTL Compiler**

### Elaboration and Synthesis

---

[iopt sequential resynthesis min effort](#)

---

## Hierarchical Flow

---

- [assemble\\_design](#) on page 386
- [generate\\_ilm](#) on page 387
- [read\\_ilm](#) on page 389

## **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 387  
[read\\_ilm](#) on page 389

## **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.

## Command Reference for Encounter RTL Compiler

### Hierarchical Flow

---

-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 386

[read\\_ilm](#) on page 389

## **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**

<code>-basename <i>string</i></code>	Specifies the directory path name and base filename for the ILM data.
<code>-logical</code>	Loads only the logical data.
<code>-module_name <i>module</i></code>	Specifies the name of the ILM.

### **Related Information**

Related commands:	<a href="#">assemble_design</a> on page 386
	<a href="#">generate_ilm</a> on page 387

## **Command Reference for Encounter RTL Compiler**

### Hierarchical Flow

---

---

## Analysis and Report

---

- [all\\_connected](#) on page 394
- [all\\_des](#) on page 395
- [all\\_des\\_inps](#) on page 396
- [all\\_des\\_insts](#) on page 397
- [all\\_des\\_outs](#) on page 398
- [all\\_des\\_seqs](#) on page 399
- [all\\_lib](#) on page 401
- [all\\_lib\\_bufs](#) on page 402
- [all\\_lib\\_ties](#) on page 403
- [analyze\\_library\\_corners](#) on page 404
- [check\\_design](#) on page 406
- [clock\\_ports](#) on page 411
- [compare\\_sdc](#) on page 412
- [create\\_timing\\_bin](#) on page 413
- [fanin](#) on page 415
- [fanout](#) on page 419
- [report](#) on page 422
- [report\\_area](#) on page 427
- [report\\_boundary\\_opto](#) on page 429
- [report\\_case\\_analysis](#) on page 431
- [report\\_cdn\\_loop\\_breaker](#) on page 432

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- [report cell delay calculation](#) on page 433
- [report clock gating](#) on page 434
- [report clocks](#) on page 440
- [report congestion](#) on page 442
- [report datapath](#) on page 443
- [report design rules](#) on page 448
- [report dft chains](#) on page 449
- [report dft\\_clock\\_domain\\_info](#) on page 454
- [report dft\\_core\\_wrapper](#) on page 455
- [report dft\\_registers](#) on page 459
- [report dft\\_setup](#) on page 463
- [report dft\\_violations](#) on page 467
- [report disabled transparent latches](#) on page 470
- [report gates](#) on page 471
- [report hierarchy](#) on page 475
- [report instance](#) on page 477
- [report low\\_power\\_cells](#) on page 480
- [report low\\_power\\_intent](#) on page 483
- [report memory](#) on page 487
- [report memory\\_cells](#) on page 488
- [report messages](#) on page 489
- [report mode](#) on page 491
- [report multibit\\_inferencing](#) on page 492
- [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

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- [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**

<i>-leaf_pin</i>	Returns only the leaf cell pins connected to the specified net.
<i>net</i>	Specifies a net. The ensuing list will return a list of all the pins connected to this net.
<i>pin</i>	Specifies a pin. The ensuing list will return the net connected to the pin.
<i>pgpin</i>	Specifies a power or ground pin. The ensuing list will return the net connected to the pin.
<i>port</i>	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**

inps	Generates a list of all input ports of the specified clock or clock domain.
insts	Generates a list of all the instances in the design.
outs	Generates a list of all output ports of the specified clock or clock domain.
seqs	Generates a list of all the sequential instances in the design.

### **Related Information**

Related commands:	<a href="#">all des inps</a> on page 396
	<a href="#">all des insts</a> on page 397
	<a href="#">all des outs</a> on page 398
	<a href="#">all des seqs</a> on page 399

## 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

<code>-clock <i>clock</i></code>	Returns a list of input ports for the specified clock or clocks.
<code>-clock_domains <i>clock_domain</i></code>	Returns a list of input ports for the specified clock domain or domains.
<code><i>design</i></code>	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

<i>design</i>	Returns a list of instances for the specified design. If a design is not specified, the instances for the current design will be returned.
<code>-unresolved</code>	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

<i>-clock</i> <i>clock</i>	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.
<i>-clock_domains</i> <i>clock_domain</i>	Returns a list of output ports for the specified clock domain or domains.
<i>design</i>	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

<code>-clock <i>clock</i></code>	Returns a list of sequential instances for the specified clock or clocks.
<code>-clock_domains <i>clock_domain</i></code>	Returns a list of sequential instances for the specified clock domain or domains.
<code>-clock_pins</code>	Returns the clock pins in the design.
<code>-data_pins</code>	Only returns a list of data pins.
<code><i>design</i></code>	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.
<code>-edge_triggered</code>	Only returns a list of flip-flops in the design.
<code>-exclude <i>instance</i></code>	Specifies a list of instances to be excluded.
<code>-inverted_output</code>	Returns sequential instances that have an inverted output (Qbar)
<code>-level_sensitive</code>	Only returns a list of latches in the design.
<code>-master_slave</code>	Returns the master slave flops.
<code>-no_hierarchy</code>	Returns sequential elements only at the top-level.
<code>-output_pins</code>	Returns the output pins in the design.
<code>-slave_clock_pins</code>	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:	<a href="#">all lib bufs</a> on page 402
	<a href="#">all lib ties</a> on page 403

## **all lib bufs**

all lib bufs

Generates a Tcl list of all the buffers in the loaded library or libraries.

### **Example**

- The following example returns all the buffers that were defined in the loaded library.

```
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.



This command should be the only command run in the synthesis session.

### **Options and Arguments**

<code>-buffer_libcell cell</code>	Specifies the libcell to be used as buffer.
<code>-cpf file</code>	Specifies the name of the CPF file that has the libraries for the multi corners.
<code>-fanin integer</code>	Specifies the number of buffers to consider in the fanin of the input pins of each libcell.  <i>Default:</i> 2
<code>-fanout integer</code>	Specifies the number of buffers to consider in the fanout of the output pins of each libcell  <i>Default:</i> 10
<code>-libraries list</code>	Specifies the list of multi-corner libraries.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### Example

The following command reads in the libraries from the CPF file. There are 8 libraries. The report shows the average delay for each library (corner) and indicates that library XS in set4 has the largest delay.

```
analyze_library_corners -cpf test.cpf
```

Library filename	Average delay
/libraries/library_domains/set8/MF	170
/libraries/library_domains/set7/MS	176
/libraries/library_domains/set6/XF	100
/libraries/library_domains/set4/XS	352
/libraries/library_domains/set2/F	162
/libraries/library_domains/set5/S	193
/libraries/library_domains/set3/VF	115
/libraries/library_domains/set1/VS	335

The slowest library : /libraries/library\_domains/set4/XS  
The average delay for this lib : 352

Libcell	Library	Delay	Slowest Library	Delay
AO22XA	/lib*/library_domains/set1/VS	354	/lib*/library_domains/set4/XS	346
AND2CSXA	/lib*/library_domains/set1/VS	283	/lib*/library_domains/set4/XS	280
BUFCSXAXA	/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.
<i>design</i>	Specifies the name of the design to write the report.
<i>file</i>	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 <code>preserve</code> attribute is set to <code>true</code> .
-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.
	<b>Note:</b> By default, the scan pins are included in all checks and reports.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

-threshold_fanout	Filters undriven or constant pins and ports with a fanout below the specified threshold.  <b>Note:</b> This option must be specified with the <code>-constant</code> option.
-through_tie_cell	Reports pins and ports connected to a constant through tie high/low cells.  <b>Note:</b> This option must be specified with the <code>-constant</code> option.
-undriven	Reports ports and pins in the design that are undriven.
-unloaded	Reports ports and sequential elements in the design that are unloaded.
-unresolved	Reports unresolved references and empty modules in the design.
-vname	Uses the Verilog names instead of RTL Compiler design hierarchy path names in the report.

## 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
      -----
      Name          Total
      -----
      Unresolved References      0
      Empty Modules            0
      Unloaded Port(s)         0
      Unloaded Sequential Pin(s) 0
      Assigns                  0
      Undriven Port(s)          0
      Undriven Leaf Pin(s)      0
      Undriven hierarchical pin(s) 1
      Multidriven Port(s)       0
      Multidriven Leaf Pin(s)    0
      Multidriven hierarchical Pin(s) 0
      Multidriven unloaded net(s) 0
      Constant Port(s)          0
      Constant Leaf Pin(s)       0
      Constant hierarchical Pin(s) 0
      Preserved leaf instance(s) 5
      Preserved hierarchical instance(s) 1
      Libcells with no LEF cell 601
      Physical (LEF) cells with no libcell 846
```

Done Checking the design.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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_SMDFFHQX8
      Total number of cell(s) with only library (.lib) info : 601

      LEF cells with no corresponding libcell
      -----
      /libraries/physical_cells/libcells/AN2D0
      ...
      /libraries/physical_cells/libcells/XOR4D4
      Total number cell(s) with only physical (LEF) Info : 846

      Done Checking the design.
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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.
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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 320

## **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**

<code>-design <i>string</i></code>	Specifies the top-level design in RTL Compiler.
<code>-detail</code>	Requests a detailed comparison report.
<code><i>file</i></code>	Specifies the file to which the report must be written.
<code>-golden_sdc <i>files</i></code>	Specifies the UNIX path to the golden (original) SDC files.
<code>-logfile <i>file</i></code>	Specifies the name of the CCD logfile. You must specify the UNIX path to the file.
<code>-netlist <i>file</i></code>	Specifies the UNIX path to the netlist. By default, the tool uses the RTL.
<code>-revised_sdc <i>files</i></code>	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.
<code>-rtl</code>	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 249

## **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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

<code>-from {pin   port   clock   external_delay   instance}</code>	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.
<code>-mode mode</code>	Specifies that the paths in the timing bin only apply for the specified mode.
<code>-name string</code>	Specifies the name of the timing bin to be created.
<code>-num_paths integer</code>	Specifies the maximum number of paths to include in the timing bin.
<code>-parent string</code>	Specifies the name of the timing bin from which you want to derive this bin.
<code>-paths string</code>	Includes the selected timing restricted paths. Create the string argument using the <u><a href="#">specify_paths</a></u> command.
<code>-slack_limit delay</code>	Includes only paths with a slack less than the specified value (in picoseconds).
<code>-through {pin   port   instance}</code>	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 <code>-through</code> 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.
<code>-to {pin   port   clock   external_delay   instance}</code>	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.
<code>-worst integer</code>	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

<b>-floating_subports</b>	Reports only floating (unconnected) supports found in the connectivity-based structural trace. You can only specify this option with the <b>-structural</b> option.  By default only non-floating supports and constants are returned.
<b>-gui</b>	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.  <b>Note:</b> The GUI should be enabled to have any effect.
<b>-max_logic_depth integer</b>	Specifies the maximum number of logic levels to go back to report the fanin cone information.  <i>Default:</i> Infinity
<b>-max_pin_depth integer</b>	Specifies the maximum number of pin levels to go back to report the fanin cone information.  <i>Default:</i> Infinity
<b>-min_logic_depth integer</b>	Specifies the minimum number of logic levels to go back to report the fanin cone information.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

*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.

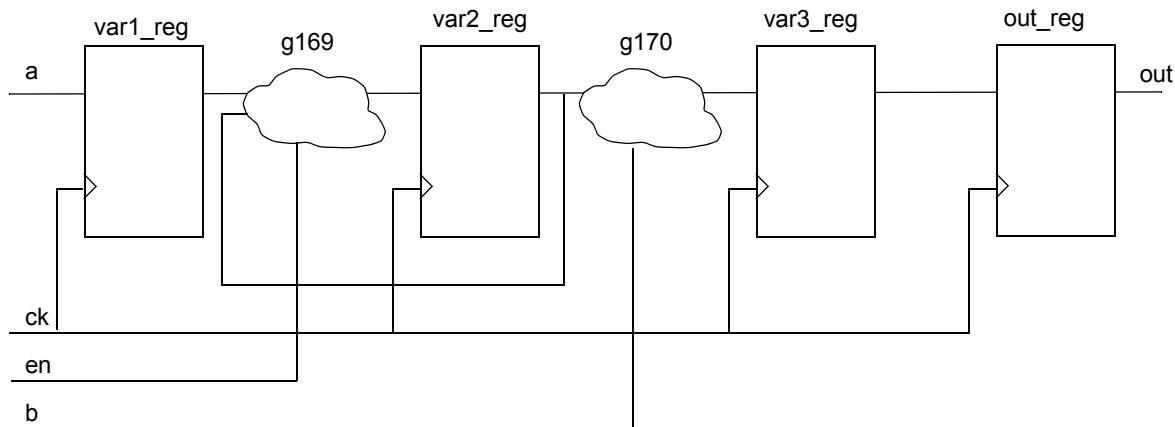
**Note:** This option does not apply to supports and constants.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

#### 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 418.

```
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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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
```

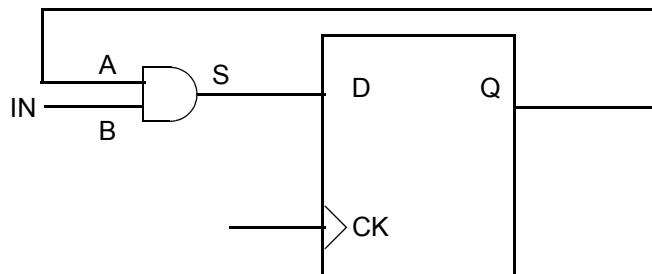
You can also use the append arrows ("`>>`").

- 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

<code>-endpoints</code>	Returns only timing end points in the fanout cone.
<code>-floating_subports</code>	Reports only floating (unconnected) subports found in the connectivity-based structural trace. You can only specify this option with the <code>-structural</code> option.  By default only non-floating subports and constants are returned.
<code>-gui</code>	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.  <b>Note:</b> The GUI should be enabled to have any effect.
<code>-max_logic_depth integer</code>	Specifies the maximum number of logic levels to go back to report the fanout cone information.  <i>Default:</i> Infinity
<code>-max_pin_depth integer</code>	Specifies the maximum number of pin levels to go back to report the fanout cone information.  <i>Default:</i> Infinity
<code>-min_logic_depth integer</code>	Specifies the minimum number of logic levels to go back to report the fanout cone information.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

*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.

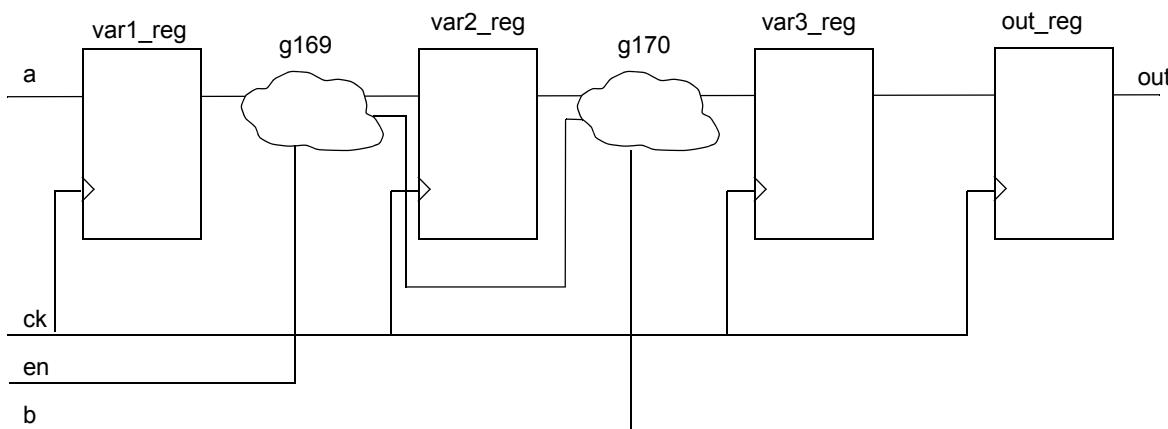
`-vname`

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.



## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
rc:/> fanout en
/designs/test/instances_comb/g169/pins_out/Z /designs/test/instances_seq/
var2_reg/pins_in/D /designs/test/instances_comb/g170/pins_in/SL /designs/
test/instances_comb/g170/pins_out/Z /designs/test/instances_seq/var3_reg/
pins_in/D
```

- The following example executes a path disable on all the endpoints to which reg1/CK fans out.

```
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.

```
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.

```
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.

```
rc:/designs/malexander/ports_in> ls -dir [fanout in1[0]] > malexander.txt
```

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

## 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`.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

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.
dft_violations	Reports the DFT violations.
disabled_transparent_latches	Reports disabled transparent latches.
gates	Generates a gates report.
hierarchy	Reports the design hierarchy information.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

instance	Generates a report on the instances of the current design.
low_power_cells	Reports information about the low power cells inserted in the design.
low_power_intent	Prints out the power intent included in the power intent file that was read in.
memory	Reports the memory usage in the compilation environment.
memory_cells	Reports the memory cells in the library.
messages	Generates a summary of error messages that have been issued.
mode	Reports the active and inactive modes of an instance.
multibit_inferencing	Reports on multibit cells in the design or library.
net_cap_calculation	Reports how the capacitance of the net is calculated.
net_delay_calculation	Reports how the net delay is calculated.
net_res_calculation	Reports how the resistance of the net is calculated.
nets	Generates a report on the nets of the current design.
opcg_equivalents	Reports the OPCG-equivalency mappings.
ple	Reports physical layout estimation data
port	Generates reports on the ports of the current design.
power	Generates a power leakage report.
power_domain	Generates a report with power domain information.
proto	Generates a report with prototype synthesis information.
qor	Generates a QoR report.
scan_compressibility	Reports the scan compressibility of a design.
sequential	Generates a report on the sequential elements of the design.
slew_calculation	Reports how the slew on the driver pin of a libcell instance is calculated.
summary	Generates an area, timing, and design rule violations report.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

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:	<a href="#">report area</a> on page 427
	<a href="#">report boundary_opto</a> on page 429
	<a href="#">report case_analysis</a> on page 431
	<a href="#">report cdn_loop_breaker</a> on page 432
	<a href="#">report cell_delay_calculation</a> on page 433
	<a href="#">report clock_gating</a> on page 434
	<a href="#">report clocks</a> on page 440
	<a href="#">report congestion</a> on page 442
	<a href="#">report datapath</a> on page 443
	<a href="#">report design_rules</a> on page 448
	<a href="#">report dft_chains</a> on page 449
	<a href="#">report dft_clock_domain_info</a> on page 454
	<a href="#">report dft_core_wrapper</a> on page 455
	<a href="#">report dft_registers</a> on page 459
	<a href="#">report dft_setup</a> on page 463
	<a href="#">report dft_violations</a> on page 467
	<a href="#">report disabled_transparent_latches</a> on page 470
	<a href="#">report gates</a> on page 471
	<a href="#">report hierarchy</a> on page 475
	<a href="#">report instance</a> on page 477
	<a href="#">report memory</a> on page 487

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

[report memory cells](#) on page 488  
[report messages](#) on page 489  
[report mode](#) on page 491  
[report multibit inferencing](#) on page 492  
[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

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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

<i>-depth integer</i>	Specifies the number of levels of recursion.
<i>design</i>	Specifies the design for which you want to generate a report. You can also cd into the particular design directory and generate the report.
<i>file</i>	Specifies the name of the file to which to write the report.
<i>-instance_hierarchy instance</i>	Reports the area of the leaf instances in the specified hierarchical instance. This option must be specified together with the -depth option.
<i>-min_count integer</i>	Specifies the minimum instance count per block.
<i>-physical</i>	Specifies to use the LEF cell width and height values to calculate the area.
<i>-summary</i>	Prints the area summary for the design.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## Examples

- The following example generates the area report for the current design.

```
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.

```
rc:/> report area -summary
=====
...
=====

Instance      Cells  Cell Area  Net Area      Wireload
-----
mult_bit_muxed_add   6       69        0 suggested_10K (S)
```

- The following command reports the area for the leaf instances of instance ma1.

```
rc:/> report area --inst_hier ma1 -depth 2
=====
...
=====

Instance      Cells  Cell Area  Net Area      Wireload
-----
ma1           3       35        0 suggested_10K (S)
g53           0       12        0 suggested_10K (S)
g52           0       12        0 suggested_10K (S)
g51           0       10        0 suggested_10K (S)

(S) = wireload was automatically selected
```

## Related Information

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 379

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
=====
...
=====
Instance   Pin           Boundary Change
-----
inst      out1  routed opposite signal through 'inst/a' around 'inst'
          out2  pushed opposite signal through 'inst/out'
```

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

#### **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**

<i>design</i>	Specifies the design for which you want to generate a report.
<i>file</i>	Reports the case constants.

### **Example**

```
-----  
report case_analysis  
-----  
Pin Name Case  
-----  
OR1/A      0  
en         1
```

### **Related Information**

Set by this command:	<a href="#">dc::set_case_analysis</a>
Set by these attributes:	(pin) <a href="#">timing_case_logic_value</a> (port) <a href="#">timing_case_logic_value</a>

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### **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**

<i>design</i>	Specifies the design for which you want the report.
<i>file</i>	Redirects the report to the specified file.
<i>-sdcfile string</i>	Creates an SDC file with the appropriate <code>set_disable_timing</code> and <code>set_false_path</code> settings.
<i>-version string</i>	Specifies a particular SDC version for the SDC file created with the <code>-sdcfile</code> option. The available versions are: 1.1, 1.3, 1.4, 1.5 or 1.7. <i>Default:</i> 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 1080

## **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**

<i>file</i>	Redirects the report to the specified file.
<i>-from_pin pin</i>	Specifies the driving pin.
<i>-from_fall</i>	Uses the fall delay calculation from the driving pin.
<i>-from_rise</i>	Uses the rise delay calculation from the driving pin.
<i>-to_fall</i>	Uses the fall delay calculation of the loading pin.
<i>-to_pin pin</i>	Specifies the loading pin.
<i>-to_rise</i>	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 have been defined through either the `define_clock` command or through the SDC constraints.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

`-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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

`-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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

## 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
                                                 aluout_reg[1]    100.000
...
                                                 aluout_reg[6]    100.000
                                                 aluout_reg[7]    100.000
-----
Total   1                         8
-----
```

8

- The following command gives the reason why the flip-flops remain ungated. In this case, there is no return value.

```
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
-----
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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
-----
  Fanout-Size      Num-CGs      Total-FFs
-----
  1 to 6          0            0
  7 to 15         2            16
  16 and higher   0            0
```

- The following command generates detailed clock-gating information for the specified clock-gating instance.

```
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/ena)
                    TCF = (0.50000, 0.114583/ns)
    test        =   LOGIC0
  Outputs:
    ck_out      =   rc_gclk
                    TCF = (0.40890, 0.317708/ns)
  Toggle Reduction = 44.40
  Gated FFs:
Module  Clock Gating Instance  Origin  Fanout  Gated Flip-flops  Register Efficiency
-----
  alu      RC_CG_HIER_INST0      RC      8      aluout_reg[0]    100.000
                                                aluout_reg[1]    100.000
...
                                                aluout_reg[6]    100.000
                                                aluout_reg[7]    100.000
-----
  Total    1                      8
```

1

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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 379](#)

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](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<code>clock</code>	Specifies the name of the clock for which you want to generate the report.
<code>file</code>	Specifies the name of the file to which to write the report.
<code>-generated</code>	Adds generated clock information to the description, uncertainty, and the relationship table.
<code>-ideal</code>	Reports a clock description with the ideal clock - clock relationship.
<code>-mode mode_name</code>	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)
Wirereload mode:       enclose
=====

Clock Description
-----
Clock                                No of
Name     Period    Rise     Fall      Domain  Pin/Port Registers
-----
CLK1    4000.0    0.0     2000.0   domain_1  Clk        5
CLK2    2000.0    0.0     1000.0   domain_1  C          0
CLK3    3000.0    0.0     1500.0   domain_2  Clk1       5
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
CLK4      6000.0  0.0   3000.0  domain_2    C1          0
Clock Network Latency / Setup Uncertainty
-----
Clock      Network Latency   Network Latency   Source Latency   Source Latency   Setup Uncertainty   Setup Uncertainty
Name       Rise             Fall            Rise           Fall          Rise            Fall
-----
CLK1      140.0           140.0          150.0         150.0        100.0          100.0
CLK2      120.0           120.0          120.0         120.0        110.0          110.0
CLK3      100.0           100.0          100.0         100.0        100.0          100.0
CLK4      0.0              0.0            0.0           0.0          0.0            0.0
Clock Relationship (with uncertainty & latency)
-----
From     To      R->R    R->F    F->R    F->F
-----
CLK1    CLK1    3900.0  1900.0  1900.0  3900.0
CLK1    CLK2    1840.0  840.0   1840.0  840.0
CLK2    CLK1    1950.0  1950.0  950.0   950.0
CLK2    CLK2    1890.0  890.0   890.0   1890.0
CLK3    CLK3    2900.0  1400.0  1400.0  2900.0
CLK3    CLK4    2800.0  2800.0  1300.0  1300.0
CLK4    CLK3    3100.0  1600.0  3100.0  1600.0
CLK4    CLK4    6000.0  3000.0  3000.0  6000.0
```

### Related Information

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 317

# Command Reference for Encounter RTL Compiler

## Analysis and Report

# 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

Item	Oflow/Avail	Gcell	(Bounding-box location)
Max. Overflow	-111/201	20,18	(480.0,432.0), (504.0,456.0)
Max. Overflow (H)	-52/118	31,29	(744.0,696.0), (768.0,720.0)
Max. Overflow (V)	-75/87	19,17	(456.0,408.0), (480.0,432.0)
Total Overflow	-7172		
Total Overflow (H)	-1329		
Total Overflow (V)	-5843		

#### **Related Information**

Affected by 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 `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.  <b>Note:</b> The mux operators are different from the MUX library cells that are picked by the mapper or are available in the technology library.
<i>design</i>	Specifies a particular design on which to report datapath operators. By default, RTL Compiler reports on the current design.
<i>file</i>	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 <i>string</i>	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 <i>Operator</i> , <i>Signedness</i> , <i>Inputs</i> , <i>Outputs</i> , <i>Cell Area</i> , <i>Line</i> , <i>Col</i> , and <i>Filename</i> .
-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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

-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 <code>synthesize -to_mapped</code> 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 <i>keys</i>	Indicates how to sort the report. You can sort on the following keys: <ul style="list-style-type: none"><li>■ <code>architectures</code> sorts the architectures in alphabetical order</li><li>■ <code>area</code> sorts by descending area</li><li>■ <code>inputs</code> sorts based on the number*width (number of bits) of the input signals—components with higher number of bits are printed first</li><li>■ <code>instances</code> sorts the instances in alphabetical order</li><li>■ <code>outputs</code> sorts based on the number*width (number of bits) of the output signals—components with higher number of bits are printed first</li><li>■ <code>operators</code> sorts by operator</li><li>■ <code>slack</code> sorts by ascending slack</li><li>■ <code>subdesigns</code> sorts the subdesigns in alphabetical order</li></ul> By default, the report does not contain the slack numbers. <b>Note:</b> You can sort on several keys.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

## Examples

- The following example generates a report of the datapath components for the rpdpl\_basic design.

```
rc:/> report datapath rpdpl_basic
=====
Module:          rpdpl_basic
Technology library: tutorial 1.0
...
=====
Instantiated datapath components

      Operator Signedness  Inputs Outputs CellArea Line Col  Filename
=====
rpdpl_basic
  d1
    u2
      module:CW_CW_multadd_builtin_wA8_wB8_wC8_wZ16
        CW/CW_multadd/builtin
          n/a   n/a       8x8x8x1 16      1239.75  38  52 impl_inf.v
+++++
  mul_1_19
    very_fast/non_booth
      *      signed      9x9     16      912.75
+++++
  add_1_24
    very_fast      +      signed     16x9     16      322.50
=====

Inferred components

      Operator Signedness  Inputs Outputs CellArea Line Col  Filename
=====
rpdpl_basic
  mul_I8_28
    module:mult_unsigned
      very_fast/non_booth
        *      unsigned    16x8     16      1044.00  18  28 impl_inf.v
=====

      Type      CellArea Percentage
-----
datapath modules  2283.75      20.55
external muxes    0.00        0.00
others            8829.00     79.45
-----
total             11112.75     100.00
```

- By default, when using the `report datapath` command on a mapped netlist containing datapath operators, you will get the area statistics of the design, as shown in the following example.

```
-----
datapath modules 4938.00      100.00
mux modules      0.00        0.00
others           0.00        0.00
-----
total            4938.00      100.00
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

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

Operator	Signedness	Inputs	Outputs	<b>CellArea</b>	Line	Col	Filename
lt_leq							
add_14_26							
module:add_signed							
very_fast	+	signed	4x4	4	<b>31.50</b>	14	26 lt_leq.v
lt_leq							
lt_13_25							
module:lt_signed							
very_fast	<	signed	4x4	1	<b>26.25</b>	13	25 lt_leq.v
lt_leq							
lt_16_25							
module:lt_unsigned							
very_fast	<	unsigned	4x4	1	<b>26.25</b>	16	25 lt_leq.v
lt_leq							
le_17_26							
module:leq_unsigned							
very_fast	<=	unsigned	4x4	1	<b>25.50</b>	17	26 lt_leq.v
-----							
Type		CellArea	Percentage				
datapath modules		109.50	75.26				
external muxes		0.00	0.00				
others		36.00	24.74				
total		145.50	100.00				

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following command sorts the report first by slack, then by area.

```
rc:/> report datapath -sort {slack area}
=====
...
=====
```

Inferred components

Operator	Signedness	Inputs	Outputs	CellArea	Line	Col	Filename	Slack
lt_leq								
lt_13_25								
module:lt_signed								
very_fast <	signed	4x4	1		26.25	13	25	lt_leq.v <b>-145.2</b>
lt_leq								
lt_16_25								
module:lt_unsigned								
very_fast <	unsigned	4x4	1		26.25	16	25	lt_leq.v <b>-145.2</b>
lt_leq								
le_17_26								
module:leq_unsigned								
very_fast <=	unsigned	4x4	1		25.50	17	26	lt_leq.v <b>-53.3</b>
lt_leq								
add_14_26								
module:add_signed								
very_fast +	signed	4x4	4		31.50	14	26	lt_leq.v <b>49.0</b>
-----								
Type	CellArea	Percentage						
datapath modules	109.50	75.26						
external muxes	0.00	0.00						
others	36.00	24.74						
total	145.50	100.00						

### Related Information

[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]... [> file]
```

Reports any design rule violations that are present in the specified design objects.

### **Options and Arguments**

<i>design</i>	Specifies the design for which you want to generate a report. By default, a report is created for all designs currently loaded in memory.
<i>file</i>	Specifies the name of the file to which to write the report.

### **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.
```

### **Related Information**

[Setting Design Rule Constraints](#) in *Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler*

Affected by this command: [synthesize](#) on page 379

Affected by these attributes: [ignore\\_library\\_max\\_fanout](#)  
[max\\_capacitance](#)  
[max\\_fanout](#)  
[max\\_transition](#)

## **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_config_mode_name`

Reports the scan chains related to the specified scan mode.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

<code>file</code>	Specifies the name of the file to which to write the report.
<code>-opcg_scan_side</code>	Reports only the side scan chains that contain the domain macro segments.
<code>-summary</code>	Condenses the scan chain report to include only chain name, scan-data pins, shift enable, clock domain and length information.

### Examples

- The following example shows one scan chain, with three user-defined segments.

```
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
-----
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following examples show the scan chains of a design before and after DFT compression.

- **Before compression**

Reporting 2 scan chains (muxed\_scan)

```
Chain 1: AutoChain_1
  scan_in:      DFT_sdi_1
  scan_out:     DFT_sdo_1
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
    bit 1        out_reg[1]  <clk (rise)>
    bit 2        out_reg[2]  <clk (rise)>
    bit 3        out_reg[3]  <clk (rise)>
    bit 4        out_reg[4]  <clk (rise)>
    bit 5        out_reg[5]  <clk (rise)>
    bit 6        out_reg[6]  <clk (rise)>
    bit 7        out_reg[7]  <clk (rise)>
    bit 8        out_reg[8]  <clk (rise)>
    bit 9        out_reg[9]  <clk (rise)>
    bit 10       out_reg[10] <clk (rise)>
-----
Chain 2: AutoChain_2
  scan_in:      DFT_sdi_2
  scan_out:     DFT_sdo_2
  shift_enable: se (active high)
  clock_domain: clk (edge: rise)
  length: 10
    bit 1        out_reg[11] <clk (rise)>
    bit 2        out_reg[12] <clk (rise)>
    bit 3        out_reg[13] <clk (rise)>
    bit 4        out_reg[14] <clk (rise)>
    bit 5        out_reg[15] <clk (rise)>
    bit 6        out_reg[16] <clk (rise)>
    bit 7        out_reg[17] <clk (rise)>
    bit 8        out_reg[18] <clk (rise)>
    bit 9        out_reg[19] <clk (rise)>
    bit 10       out_reg[20] <clk (rise)>
```

- **After compression**

The report shows in addition

- How the original scan chains have been divided in several internal chains.
- For each internal chain the START and END (separate scan data input and output) are given together with the length of the internal channel.
- An additional scan chain, mask\_chain (if the user opted to add making logic)

The mask chain is comprised of abstract segments only.

The shift-enable signal of the mask chain is reported as NONE because it is a *connected* shift enable.

# Command Reference for Encounter RTL Compiler

## Analysis and Report

```
Reporting 3 scan chains (muxed_scan)

Chain 1: AutoChain_1 (compressed)
scan_in: DFT_sdi_1
scan_out: DFT_sdo_1
shift_enable: se (active high)
clock_domain: clk (edge: rise)
length: 10
    <START compressed internal chain AutoChain_1_0 (sdi: g121/SWBOX_SI[0])>
    bit 1      out_reg[1]  <clk (rise)>
    bit 2      out_reg[2]  <clk (rise)>
    <END   compressed internal chain AutoChain_1_0 (sdo: g121/SWBOX_SO[0])
(length: 2)>
    <START compressed internal chain AutoChain_1_1 (sdi: g121/SWBOX_SI[1])>
    bit 3      out_reg[3]  <clk (rise)>
    bit 4      out_reg[4]  <clk (rise)>
    <END   compressed internal chain AutoChain_1_1 (sdo: g121/
SWBOX_SO[1]) (length: 2)>
    <START compressed internal chain AutoChain_1_2 (sdi: g121/SWBOX_SI[2])>
    bit 5      out_reg[5]  <clk (rise)>
    bit 6      out_reg[6]  <clk (rise)>
    <END   compressed internal chain AutoChain_1_2 (sdo: g121/SWBOX_SO[2])
(length: 2)>
    <START compressed internal chain AutoChain_1_3 (sdi: g121/SWBOX_SI[3])>
    bit 7      out_reg[7]  <clk (rise)>
    bit 8      out_reg[8]  <clk (rise)>
    <END   compressed internal chain AutoChain_1_3 (sdo: g121/SWBOX_SO[3])
(length: 2)>
    <START compressed internal chain AutoChain_1_4 (sdi: g121/SWBOX_SI[4])>
    bit 9      out_reg[9]  <clk (rise)>
    bit 10     out_reg[10] <clk (rise)>
    <END   compressed internal chain AutoChain_1_4 (sdo: g121/SWBOX_SO[4])
(length: 2)>
-----
Chain 2: AutoChain_2 (compressed)
scan_in: DFT_sdi_2
scan_out: DFT_sdo_2
shift_enable: se (active high)
clock_domain: clk (edge: rise)
length: 10
    <START compressed internal chain AutoChain_2_5 (sdi: g121/SWBOX_SI[5])>
    bit 1      out_reg[11] <clk (rise)>
    bit 2      out_reg[12] <clk (rise)>
    <END   compressed internal chain AutoChain_2_5 (sdo: g121/SWBOX_SO[5])
(length: 2)>
...
...
    <START compressed internal chain AutoChain_2_9 (sdi: g121/SWBOX_SI[9])>
    bit 9      out_reg[19] <clk (rise)>
    bit 10     out_reg[20] <clk (rise)>
    <END   compressed internal chain AutoChain_2_9 (sdo: g121/SWBOX_SO[9])
(length: 2)>
-----
Warning - could not find shift_enable signal for chain /designs/top/dft/
report/actual_scan_chains/mask_chain
Chain 3: mask_chain
scan_in: msi
scan_out: mso
shift_enable: NONE
clock_domain: mck (edge: rise)
length: 10
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
START segment DFT_segment_1 (type: abstract)
  # @ bit 1, length: 4
  pin      g121/msi[0] <mck (rise)>
  pin      g121/mso[0] <mck (rise)>
END segment DFT_segment_1
START segment DFT_segment_3 (type: abstract)
  # @ bit 5, length: 3
  pin      g121/msi[2] <mck (rise)>
  pin      g121/mso[2] <mck (rise)>
END segment DFT_segment_3
START segment DFT_segment_2 (type: abstract)
  # @ bit 8, length: 3
  pin      g121/msi[1] <mck (rise)>
  pin      g121/mso[1] <mck (rise)>
END segment DFT_segment_2
-----
```

- The following command reports two side scan chains, each with one OPCG segment.

```
rc:/> report dft_chains -opcg_side_scan

Reporting 2 side scan chains
Side-scan chain 1:
  scan_in:    io[4] (hookup: p/p4/Y)
  clock domain: opcg_load_clk
  length: 7
    START segment OPCG_DOMAIN_SEG_1 (type: abstract)
      #@bit 1, length 7
      pin      DFT_OD2/PGMSI
      pin      DFT_OD2/PGMSO
    END segment OPCG_DOMAIN_SEG_1
-----
Side-scan chain 2:
  scan_in:    DFT_sdi_1
  clock domain: opcg_load_clk
  length: 6
    START segment OPCG_DOMAIN_SEG_0 (type: abstract)
      #@bit 1, length 6
      pin      DFT_OD1/PGMSI
      pin      DFT_OD1/PGMSO
    END segment OPCG_DOMAIN_SEG_0
-----
```

### Related Information

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**

<i>design</i>	Specifies the design for which you want to report the DFT clock domain information.
<i>-endpoint_limit integer</i>	Prevents reporting endpoints if the number is less than the specified limit.
<i>-report_internal</i>	Reports the number of endpoints that receive data from the same clock domain
<i>-system_clock_violations</i>	Reports only system clock violations.
<i>-verbose</i>	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 823

## **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.

# Command Reference for Encounter RTL Compiler

## Analysis and Report

---

### Examples

- The following examples illustrate the different report options on the same design.

```
rc:/> report dft_core_wrapper test
Reporting 13 ports/pins
Port/Pin          type      usage      wrapper_objects
=====  ======  ======  =====
WEXT            wrapper_control -          -
WIG             wrapper_control -          -
WINT            wrapper_control -          -
WOG             wrapper_control -          -
WSEN_in         wrapper_control -          -
clk             wrapper_control -          -
WSEN_out        shift_enable   -          -
se              shift_enable   -          -
in1             shared       input       wrap_1
                shared       output      wrap_4
in2             shared       input       wrap_3
                shared       input       wrap_1
in2             shared       output      wrap_4
in3             shared       input       wrap_3
out1            shared       input       wrap_2
                shared       input       wrap_1
                shared       input       wrap_4
out2            shared       input       wrap_1
                shared       input       wrap_4

Wrapper insertion summary report
=====
Number of Dedicated wrapper cells :0      input bounding :0      output bounding :0
Number of Shared wrapper cells :4       input bounding :3      output bounding :1
-----
Total no. of wrapper cells      :4      input bounding :3      output bounding :1

rc:/> report dft_core_wrapper test -wrapped
Reporting 13 ports/pins
Port/Pin          type      usage      wrapper_objects
=====  ======  ======  =====
in1             shared       input       wrap_1
                shared       output      wrap_4
in1             shared       input       wrap_3
in2             shared       input       wrap_1
                shared       output      wrap_4
in2             shared       input       wrap_3
in3             shared       input       wrap_2
out1            shared       input       wrap_1
                shared       input       wrap_4
out2            shared       input       wrap_1
                shared       input       wrap_4

Wrapper insertion summary report
=====
Number of Dedicated wrapper cells :0      input bounding :0      output bounding :0
Number of Shared wrapper cells :4       input bounding :3      output bounding :1
-----
Total no. of wrapper cells      :4      input bounding :3      output bounding :1

rc:/> report dft_core_wrapper -report_flops
Reporting 13 ports/pins
Port/Pin          type      Usage      wrapper_objects      wrapper_flops
-----  -----  -----  -----  -----
WEXT            wrapper_control -          -
WIG             wrapper_control -          -
WINT            wrapper_control -          -
WOG             wrapper_control -          -
WSEN_in         wrapper_control -          -
clk             wrapper_control -          -
WSEN_out        shift_enable   -          -
se              shift_enable   -          -
in1             shared       input       wrap_1          tmp5_reg
                shared       input       wrap_4          tmp4_reg
in1             shared       output      wrap_3          tmp3_reg
```

# Command Reference for Encounter RTL Compiler

## Analysis and Report

---

```

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:**

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/clk	excluded	-	-
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:**

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/clk	excluded	-	-
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

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
i_core1/in[3]           shared      input      wrap_in_2
i_core1/out[0]          dedicated   output     wrap_out_4
i_core1/out[1]          dedicated   output     wrap_out_3
i_core1/out[2]          dedicated   output     wrap_out_2
i_core1/out[3]          dedicated   output     wrap_out_1

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 857

## **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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### Options and Arguments

<i>design</i>	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
<code>-dont_scan</code>	Reports all edge-triggered registers that are not to be mapped to scan flops, or connected into the top-level chains.
<code>-fail_tdrc</code>	Reports all edge-triggered registers that fail the DFT rule checks.
<i>file</i>	Specifies the name of the file to which to write the report.
<code>-latch</code>	Reports all registers of type latch in the design.  <b>Note:</b> Latch instances which are instantiated as lockup elements in a preserved segment are reported with the <code>-lockup</code> option.
<code>-lockup</code>	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.
<code>-misc</code>	Reports all registers that are not reported through any of the other options, such as clock-gating registers.
<code>-multi_bit</code>	Reports all registers that are implemented as multibit elements.
<code>-pass_tdrc</code>	Reports all edge-triggered registers that pass the DFT rule checks.
<code>-shift_reg</code>	Reports all registers that are part of shift register segments.
<code>-test_clock test_clock</code>	Returns the number of registers controlled by the specified test clock.
<code>-test_clock_edge test_clock_edge</code>	Returns the number of registers controlled by the specified test clock edge.  <b>Note:</b> This option must be specified with the <code>-test_clock</code> option.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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
  ...
  out2_reg_0    PASS; Test clock: test_clk2/rise
  ...
  out2_reg_4    PASS; Test clock: test_clk2/fall
  ...
  out3_reg_0    PASS; Test clock: test_clk3/rise
  ...
  out3_reg_4    PASS; Test clock: test_clk3/fall
  ...

Reporting registers that fail DFT rules
Reporting registers that are preserved or marked dont-scan
Reporting registers that are marked Abstract Segment Dont Scan
Reporting registers that are part of shift register segments
Reporting registers that are identified as lockup elements
DFT_lockup_g1
DFT_lockup_g348
DFT_lockup_g349
DFT_lockup_g350
Reporting registers that are level-sensitive elements
Reporting misc. non-scan registers
Summary:
Total registers that pass DFT rules: 27
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
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](#)

# Command Reference for Encounter RTL Compiler

## Analysis and Report

## 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

<i>design</i>	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
<i>file</i>	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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
=====
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
=====
    top

Scan Style
=====
    muxed_scan
Design has a valid DFT rule check status

Global Constraints
=====
    Minimum no of Scan chains: no_value
    Maximum length of scan chains: no_value
    Lock-up element type: level_sensitive
    Mix clock edges in scan chain: true
    Prefix to name user defined scan chains: DFT_

Test signal objects
=====
    shift_enable:
        object name: SE
        pin name: SE
        hookup_pin: SE
        hookup_polarity: non_inverted
        active: high
        ideal: true
        user defined: true

Test clock objects
=====
    test_clock:
        object name: clk
        user defined: false
        source: clk
        root source: clk
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
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]
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
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 761

[define\\_dft\\_test\\_mode](#) on page 765

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**

<code>-abstract</code>	Reports all abstract segment test mode violations.
<code>-async</code>	Reports all async set and async reset violations.
<code>-clock</code>	Reports all clock violations.
<code>design</code>	Specifies the design for which you want to generate a report. By default a report is created for all designs currently loaded in memory.
<code>file</code>	Specifies the name of the file to which to write the report.
<code>-race</code>	Specifies to report potential race condition violations.
<code>-shiftreg</code>	Reports all shift register segment violations.
<code>-tristate</code>	Reports tristate design rules checking violations.
<code>-xsource</code>	Reports X-Source violations.
<code>-xsource_by_instance</code>	Reports X-Source violations by instance.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### Examples

- The following example shows that the design has four violations, but only detailed information on the clock violations is requested.

```
rc:/> report dftViolations -clock
Total 4 violations (muxed_scan)

Clock Rule Violations
=====
Reporting 2 clock violations

Violation #0:
  Object name: vid_0_clock
  Type: clock violation
  Description: [CLOCK-05] internal or gated clock signal
  Source: g1/z (test10.v:12)
  Number of registers affected: 4
  Affected registers:
    out1_reg[0]
    out1_reg[1]
    out1_reg[2]
    out1_reg[3]

Violation #1:
  Object name: vid_1_clock
  Type: clock violation
  Description: [CLOCK-06] clock signal driven by a sequential element
  Source: divClk_reg/q (test10.v:14)
  Number of registers affected: 4
  Affected registers:
    out2_reg[0]
    out2_reg[1]
    out2_reg[2]
    out2_reg[3]

Violation Rule Summary Report
=====
[CLOCK-05] internal or gated clock signal : 1
[CLOCK-06] clock signal driven by a sequential element : 1
```

- The following example reports one tristate net contention violation.

```
rc:/> report dftViolations
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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
...
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 771

Related attributes: [Violations Attributes](#)

# Command Reference for Encounter RTL Compiler

## Analysis and Report

## **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 80

# Command Reference for Encounter RTL Compiler

## Analysis and Report

# report gates

```
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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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
-----
flopdrs 9 72.000 tutorial
inv1 35 105.000 tutorial
nand2 112 112.000 tutorial
nor2 37 55.500 tutorial
xor2 17 34.000 tutorial
-----
total 210 378.500
-----
Type Instances Area Area %
-----
sequential 9 72.000 19.0
inverter 35 105.000 27.7
logic 166 201.500 53.2
-----
total 210 378.500 100.0
```

- The following example shows the gate report for a multibit design. In this case, the tool uses the (++) annotation to mark the multibit cells. The rest of the report does not change.

```
Gate Instances Area Library
-----
DUAL_DEFQD (++) 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
```

## Command Reference for Encounter RTL Compiler

### Analysis and 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
=====
...
=====

          Leakage      Internal
      Gate    Instances     Area    Power (nW)    Power (nW)   Library
-----
ACHCONX2TH           1    28.856      59.948    372.198   slow_hvt
ADDHX1               1    39.040      158.364    276.327   slow
...
XOR2XL              10   118.820      339.664    471.216   slow
XOR2XLTH            449  5335.018     6259.161   22146.660  slow_hvt
XOR3XL              1    28.856      83.655     80.446   slow
-----
total                11980 109000.238 139743.990 505460.121

          Leakage      Leakage      Internal      Internal
      Library Instances Instances % Power (nW) Power % Power (nW) Power %
-----
slow                 1919       16.0  74607.161   53.4  168629.031   33.4
slow_hvt            10061      84.0  65136.830   46.6  336831.090   66.6
-----
          Leakage      Leakage      Internal      Internal
      Type    Instances     Area    Area % Power (nW) Power % Power (nW) Power %
-----
sequential           212    7864.054    7.2  17285.068   12.4  112374.696   22.2
inverter             1684   5961.269    5.5  5752.468    4.1  22492.255   4.4
buffer               37     364.941    0.3  1877.474    1.3  3920.823   0.8
logic                10047  94809.9748   87.0 114828.979   82.2  366672.348   72.5
-----
total                11980 109000.2382 100.0 139743.990 100.0 505460.121   100.0
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following example reports the yield cost and yield percentage values for each library cell.

```
Gate Instances Area Cost Yield Library
-----
flopdrs      33  264.000 3.39278e-06 99.9997 tutorial
inv1         103 51.500  1.5022e-06 99.9998 tutorial
nand2        315 315.000 1.08311e-05 99.9989 tutorial
nor2          19  28.500  6.79989e-07 99.9999 tutorial
-----
total        470 659.000 1.64061e-05 99.9984

Type Instances Area Area %
-----
sequential    33  264.000 40.1
inverter     103 51.500  7.8
logic         334 343.500 52.1
-----
total        470 659.000 100.0
```

### Related Information

[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 379

# Command Reference for Encounter RTL Compiler

## Analysis and Report

## 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 )

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

#### **Related Information**

Affected by this attribute: [hdl\\_track\\_filename\\_row\\_col](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<i>file</i>	Specifies the name of the file to which to write the report.
<code>-detail</code>	Reports information such as pin direction, propagated constants, propagated pins, and disabled timing arcs. Disabled timing arc info is only available for mapped designs.
<i>instance</i>	Specifies the leaf instance for which to generate the report.
<code>-power</code>	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 (*).  <b>Note:</b> The <code>lp_power_unit</code> attribute does not affect the units of the Arc Energy column.
<code>-timing</code>	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:          dpldalg
...
=====
Instance Timing Info
-----
Instance n0000D3666 of libcell nor2

Arc  Arc   Slew      Slew
From To    In    Load   Out   Delay   Slack
-----
A r  Y f   46.2  20.7  57.2  136.0 -1022.4
A f  Y r   46.2  20.7  57.2  136.0 -1022.4
B r  Y f   16.5  20.7  57.2  134.0 -900.2
B f  Y r   16.5  20.7  57.2  134.0 -900.2
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following example shows the timing and power information for instance o\_m2\_clk2\_1\_reg\_0.

```
=====
...
Module: m1
Technology library: slow 1.0
Operating conditions: slow (balanced_tree)
Wireload mode: enclosed
=====
Instance m2/o_m2_clk2_1_reg_0 of libcell EDFFX1

Arc Arc Slew      Slew
From To In Load Out Delay Slack
-----
CK   Q  0.0  10.0  173.9  387.0  inf
CK   QN 0.0    0.0    64.3      inf

Instance Power Info
-----
Instance m2/o_m2_clk2_1_reg_0 of Libcell EDFFX1

Leakage Internal Net
Power(nW) Power(nW) Power(nW)
-----
32.822   20652.925   182.250

Computed          Computed          Net
Pin     Net       Probability      Toggle Rate(/ns) Power(nW)
-----
Q      o_m2_clk2_1[0]        0.6        0.0312    182.250
CK     n_0            0.5        1.8724    4913.916
D      in_2[21]         0.5        0.0208    138.510
E      en_2[7]          0.5        0.0208    473.850

Arc Arc          Arc          Arc
From To When Activity(/ns) Energy(fJ)
-----
CK   CK  !D & !E      0.4681    9.396
CK   CK  D & !E      0.4681    9.563
CK   CK  !D & E       0.4681   11.860
CK   CK  D & E       0.4681   11.973
D    D           0.0208    7.726
CK   CK  D & E       0.4681   11.973
CK   Q            0.0312    7.902
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following example shows a detailed instance report.

```
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
TE    in          0
Q     out         clk(+)

Disabled Arcs by constant propagation
-----
/libraries/mylibraries/libcells/mycell/inarcs/CP_TE_Ha0
/libraries/mylibraries/libcells/mycell/TE/inarcs7CP_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 <i>design</i>	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.
<i>file</i>	Specifies the name of the file to which to write the report.
-instance <i>instance</i>	Restricts the reported information to the specified hierarchical instance.
-isolation_only	Reports isolation cells only
-level_shifter_only	Reports level shifters only.
-lp_instance <i>instance</i>	Reports detailed information for the specified low power cell instance.
-rule { <i>isolation_rule</i>   <i>level_shifter_rule</i>   <i>state_retention_rule</i> }	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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

## 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
  u1/in                  (pd1 -> pd1)  pd2_ls_self_hiconn_0 (LVLHLD8BWPHVT)
  u1/out                 (pd1 -> pd2)  pd2_ls_self_hiconn_1 (LVLHLD8BWPHVT)
-----
Total Level Shifter:      2
=====
```

# Command Reference for Encounter RTL Compiler

## Analysis and Report

- 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        : LVLHLD8BWPHT (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 1016  
[commit\\_power\\_intent](#) on page 1028

## **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 <i>design</i>	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 <i>domain</i>	Reports all the rules defined for the specified domain.
<i>file</i>	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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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 precede-
nce
will be applicable.
Domain : SW
ISO-Rule   : 9
=====
```

- The following command requests a detailed report. The report prints the details of the rules.

```
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
....
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following report gives the summary of the power intent file.

```
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 1016  
[commit\\_power\\_intent](#) on page 1028  
[read\\_power\\_intent](#) on page 1030

# Command Reference for Encounter RTL Compiler

## Analysis and Report

# 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.

# Command Reference for Encounter RTL Compiler

## Analysis and 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:

```

=====
# Generated by: Cadence Encounter(R) RTL Compiler 10.1.100
# Generated on: Feb 23 2010 09:34:16
=====

      Library : RF32X32.lib
      Memory Cell: RF32X32
      Memory Type: ram
      Memory address width: 5
      Memory word width: 32

-----
```

PIN/BUS	DIRECTION	IS_BUS	BUS_TYPE	CAPACITANCE	RELATED_PIN	TIMING_TYPE	TIMING_SENSE	
Q	output	31 to 0	YES	RF32X32_DATA		CLK	rising_edge	non_unate
D	input	31 to 0	YES	RF32X32_DATA	0.002	CLK CLK	hold_rising setup_rising	
CEN	input	0 to 0	NO		0.002	CLK CLK	hold_rising setup_rising	
WEN	input	0 to 0	NO		0.010	CLK CLK	hold_rising setup_rising	
A	input	4 to 0	YES	RF32X32_ADDRESS	0.008	CLK CLK	hold_rising setup_rising	

---

CLOCK	DIRECTION	CAPACITANCE	MAX_TRANSITION	MIN_PULSE_WIDTH_HIGH	MIN_PULSE_WIDTH_LOW	MIN_PERIOD
CLK	input	0.038	1.000	0.055	0.130	0.867

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<code>-all</code>	Reports all messages since you started this RTL Compiler run.
<code>-error</code>	Reports the error messages.
<code>file</code>	Specifies the name of the file to which to write the report.
<code>-include_suppressed</code>	Reports those messages that were suppressed through the <code>suppress_messages</code> command.
<code>-info</code>	Reports the information messages.
<code>-warning</code>	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.

```
rc:/> report messages
=====
Message Summary
=====

Num  Sev      Id          Message Text
-----
1  Info  ELAB-VLOG-9  Variable has no fanout.
                           This variable is not driving anything and will be
                           simplified
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
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](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<code>file</code>	Specifies the name of the file to which to write the report.
<code>instance</code>	Specifies the instance for which to report the mode information,

### Example

The following command shows the mode information for instance `i1`.

```
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**

```
report multibit_inferencing [-comb] [-seq] [-power]
    [[-instance_hier instance] [design]] | -lib
    [-not_merged_summary] [-not_merged_instance_list]
    [-no_header] > file
```

Reports detailed information on the combinational and sequential multibit cells that are used in the design or available in the libraries. Use the options to filter the information. When no options are specified, the report returns the information for all multibit cells used in the design.

### **Options and Arguments**

<code>-comb</code>	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.
<code>design</code>	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.  <b>Note:</b> You cannot specify this option together with the <code>-lib</code> option.
<code>file</code>	Redirects the report to the specified file.
<code>-instance_hier <i>instance</i></code>	Prints the report for the specified hierarchical instance.
<code>-lib</code>	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.  <code>Multibit_Usable</code> shows whether a libcell is valid to use during multibit merging or not.  <b>Note:</b> This option can be used after the libraries are loaded and before the design is loaded.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

**Note:** You cannot specify this option together with the `-instance_hier` option or when you specify the design.

<code>-no_header</code>	Prints the report without a header.
<code>-not_merged_instance_list</code>	Prints all single bit sequential instances that could not be merged into multobit instances.
<code>-not_merged_summary</code>	Prints for each of the different reasons the number of instances that were not merged into a multibit instance.
<code>-power</code>	Reports leakage power information for the multibit cells in the design.
<code>-seq</code>	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
```

```
=====
Generated by:          Encounter(R) RTL Compiler version
Generated on:          date
Technology libraries: XYZ_primitive 1.005489
                      XYZ_multibit 0.000001
Operating conditions: WORST_TREE (worst_case_tree)
Wireload mode:         enclosed
Area mode:            timing library
=====

Library : Combinational Multibit Libcells info
=====

Comb_Mbit libcell  Avoid  Bitwidth  Leakage Power  Area           Library           Library Domain
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
AND2_2B      false    2        1.23     1.10          XYZ_multibit   NA
                                                              
Seq_Mbit libcell  Avoid  Multibit_Usable  Bitwidth  Leakage Power  Area           Library           Library Domain
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
DFD_1_2B      false    true       2        14.32    7.53          XYZ_multibit   lp_domain
DFD_1_4B      false    true       4        14.32    15.05         XYZ_multibit   lp_domain
DFD_1_8B      false    false      8        14.32    30.11         XYZ_multibit   lp_domain
.
DFD_1_2B      false    true       2        14.32    7.53          XYZ_multibit   gp_domain
DFD_1_4B      false    true       4        14.32    15.05         XYZ_multibit   gp_domain
DFD_1_8B      false    false      8        14.32    30.11         XYZ_multibit   gp_domain
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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

Comb_Mbit libcell Bitwidth Count Total Area Multibit Conversion % Library
-----
AND2_2B           2      1     1.10    100.00 XYZ_multibit
-----
Total             1     1.10    100.00
```

- The following command reports the combinational multibit cells used in the design with their leakage power information.

```
rc:/> report multibit_inferencing -comb -power
=====
...
=====

Combinational Multibit cells usage statistics
=====
Total Combinational instances merged: 2
Total Combinational instances not merged: 0

Comb_Mbit libcell Bitwidth Count Total Area   Total Power(nW) Multibit Conversion % Library
-----
AND2_2B           2      1     1.10      1.23        100.00 XYZ_multibit
-----
Total             1     1.10      1.23        100.00
```

- 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
=====

Merged          Not Merged       Multibit Conversion %
-----
All Sequentials    14            0        100.00
-FlipFlops        14            0        100.00
-Latches          0            0         0.00
-----

Seq_Mbit libcell Bitwidth Count Total Area Multibit Conversion % Library
-----
DFF_1_2B (F)      2      1     7.53    14.29 XYZ_multibit
DFF_1_4B (F)      4      1     15.05   28.57 XYZ_multibit
DFF_1_8B (F)      8      1     30.11   57.14 XYX_multibit
-----
Total             3     52.69    100.00
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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 incrementaloptimization.

Sequential Multibit not merged statistics
=====
Total Sequential instances not merged: 1318

Reason of not merging          Sequential Instances          Description
-----
INST_NO_VALID_REPLACEMENT     1312      No valid replacement /No valid multibit libcell present
INST_SPLIT                     5          Single bit created after splitting the multibit
INST_WORSE_QOR                 1          Move not accepted as it is giving worse QoR
```

- The following command lists the instances that did not get merged with the reason.

```
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]
-----
...
...
```

### Related Information

[Mapping to Multibit Cells in \*Encounter RTL Compiler Synthesis Flows\*](#)

## **report net\_cap\_calculation**

```
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)

From      To          Wire res   Wire cap Fanout Pin cap of  Total pin cap of  Net
pin       pin        of net     of net of net of net to pin      all to pins  delay
-----
in1[0]    inst1/g43/A  0.000     7.2      1       5.3            5.3        0.0
```

### **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

```
report nets [-hierarchical] [-pin pin...]
            [-minfanout integer] [-maxfanout integer]
            [net | instance]... [-sort string]
            [-cap_worst integer] [> file]
```

Generates a report on the nets of the current design. The report gives information for the top-level nets in the design. You can specify pin names, nets, instances, maximum and minimum fanout threshold values, nets, and instances. Control the data printed out using the `-minfanout` and `-maxfanout` options for nets that have fanout between these values. Nets that are followed by the "@" symbol in parenthesis indicate SPEF annotation.

### Options and Arguments

<code>-cap_worst <i>integer</i></code>	Specifies the number of worst capacitance nets that are to be reported.
<code><i>file</i></code>	Specifies the name of the file to which to write the report.
<code>-hierarchical</code>	Reports all the nets in the design hierarchy.
<code>-maxfanout</code>	Specifies an <i>integer</i> value and reports nets whose fanouts are below the given threshold value.
<code>-minfanout</code>	Specifies an <i>integer</i> value and reports nets whose fanouts are above the given threshold value.
<code><i>net</i>   <i>instance</i></code>	Reports information on the specified nets or nets belonging to the instance.
<code>-pin <i>pin</i></code>	Specifies a list of pin names and reports the nets connected to the pins.
<code>-sort</code>	Specifies the field name on which to sort. Valid field names are <code>load</code> , <code>resistance</code> , or <code>capacitance</code> .

### 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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following example reports the five worst capacitance nets in the top level.

```
rc:\> report nets -cap_worst 5
=====
Generated by:          RTL Compiler (RC) Version
Generated on:          Date
Module:                m1
Technology library:   slow 1.5
Operating conditions: slow (balanced_tree)
Wireload mode:        segmented
=====

          Wire      Wire      Wireload
Net    Loads   Drivers Cap(fF) Res(k-ohm) Model
-----
ai      2       3       14.5    0.000
n_135  2       2       10.4    0.000
n_0     2       1       6.7     0.000
ao[5]   1       2       6.7     0.000
bi      1       1       3.6     0.000
=====
```

- The following example sorts the nets in the top level by the number of loads.

```
rc:\> report nets -sort load
=====
...
=====
          Wire      Wire      Wireload
Net    Loads   Drivers Cap(fF) Res(k-ohm) Model
-----
n_1    8       1
c\k    3       1       0.0     0.000
n_135 2       2       10.4    0.000
n_0    2       1       6.7     0.000
ai     2       3       14.5    0.000
.....
=====
```

- The following example provides specific information on the n\_0 ai net.

```
rc:\> report net n_0 ai
=====
...
=====
      Total    Slew    Slew
Net Cap(fF)  Rise Fall Driver(s)  Load(s)
-----
n_0    20.3   0.0   0.0   g24/Y   g23/A
                                g22/A
                                bx_reg3/CK
ai     12.0   0.0   0.0   ai      g25/A
                                bx_reg2/D
=====
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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
=====
...
=====
```

Net	Loads	Drivers	Wire Cap(fF)	Wire Res(k-ohm)	Wireload Model
in1a[0]	1	1	0.4	0.000	AL_SMALL
in1a[1]	1	1	0.4	0.000	AL_SMALL
in1b[0]	1	1	0.4	0.000	AL_SMALL
in1b[1]	1	1	0.4	0.000	AL_SMALL
out2a[0]	1	1	0.4	0.000	AL_SMALL
out2a[1]	1	1	0.4	0.000	AL_SMALL
out2b[0]	1	1	0.4	0.000	AL_SMALL
out2b[1]	1	1	0.4	0.000	AL_SMALL
out3a[0]	1	1	0.0	0.000	AL_SMALL
out3a[1]	1	1	0.0	0.000	AL_SMALL
out3b[0]	1	1	0.0	0.000	AL_SMALL
out3b[1]	1	1	0.0	0.000	AL_SMALL
out4a[0]	1	1	0.0	0.000	AL_SMALL
out4a[1]	1	1	0.0	0.000	AL_SMALL
out4b[0]	1	1	0.0	0.000	AL_SMALL
out4b[1]	1	1	0.0	0.000	AL_SMALL

```
=====
```

You can specify an instance name to the above example and get information on the nets associated with the instance(s) whose fanout is less than 2.

- The following example reports the nets associated with the g22/A bx\_reg2/D pin.

```
rc:/> report net -pin "g22/A bx_reg2/D"
=====
...
=====
```

Net	Total Cap(fF)	Slew Rise	Slew Fall	Driver(s)	Load(s)
n_0	20.3	0.0	0.0	g24/Y	g23/A g22/A bx_reg3/CK
ai	12.0	0.0	0.0	ai	g25/A bx_reg2/D

```
=====
```

### 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

<i>design</i>	Specifies the design for which you want to report the OPCG-equivalency mappings.
<i>file</i>	Specifies the name of the file to which to write the report.
<code>-pinVal</code>	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.

```
rc:/> report opcg_equivalents  
OPCG equivalent table contains 2 entries {  
{  
    Scan cell: /libraries/myspecial/libcells/SDFFQN_X1M  
    Opcg cell: /libraries/myspecial/libcells/S2DFFQQN_X1M  
    Edge_mode: TEL  
    Loop_back: TI  
    Pin map: {SE SE} {SI SI} {D D} {CK CK} {QN QN}  
}  
  
{  
    Scan cell: /libraries/myspecial/libcells/SDFFQ_X1M  
    Opcg cell: /libraries/myspecial/libcells/S2DFFQQN_X1M  
    Edge_mode: TEL  
    Loop_back: TI  
    Pin map: {D D} {CK CK} {Q Q} {SE SE} {SI SI}  
}  
}
```

### Related Information

Affected by these commands:      [reset\\_opcg\\_equivalent](#) on page 890  
                                        [set\\_opcg\\_equivalent](#) on page 894

Related commands:      [replace\\_opcg\\_scan](#) on page 878

# Command Reference for Encounter RTL Compiler

## Analysis and Report

# 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
                      tpz973g 230
                      pllclk 4.3
                      ram_128x16A 1.1
                      ram_256x16A 1.1
                      rom_512x16A 1.1
                      physical_cells
Operating conditions: slow
Interconnect mode:   global
Area mode:           physical library
=====

Aspect ratio       : 1.00
Shrink factor     : 1.00
Scale of res/length : 1.00
Scale of cap/length : 1.00
Net derating factor : 1.00
Site size          : 5.70 um (from lef [tech+cell])
```

Layer Name	Direction	Utilization	Capacitance / Length (pF/micron)	Data source: cap_table_file
M1	H	1.00	0.000274	
M2	V	1.00	0.000242	
M3	H	1.00	0.000242	
M4	V	1.00	0.000242	
M5	H	1.00	0.000242	

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

M6 V 1.00 0.000304

Layer Name	Direction	Utilization	Resistance / Length (ohm/micron)	Data source: lef_library
Metal1	H	1.00	0.439130	
Metal2	V	1.00	0.360714	
Metal3	H	1.00	0.360714	
Metal4	V	1.00	0.360714	
Metal5	H	1.00	0.360714	
Metal6	V	1.00	0.102273	

Layer Name	Direction	Utilization	Area / Length (micron)	Data source: lef_library
Metal1	H	1.00	0.230000	
Metal2	V	1.00	0.280000	
Metal3	H	1.00	0.280000	
Metal4	V	1.00	0.280000	
Metal5	H	1.00	0.280000	
Metal6	V	1.00	0.440000	

### Related Information

“Checking the Physical Layout Estimation Information” in [Simple PLE Flow](#), [Spatial Flow](#), and [RC-P Flow](#) in *Design with RTL Compiler Physical*

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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

<i>-delay</i>	Reports external delay information (rise and fall delay and the external delay object)
<i>-driver</i>	Reports the external driver name and the slew (rise and fall) values of the ports.
<i>file</i>	Specifies the name of the file to which to write the report.
<i>-load</i>	Reports the external fanout load and the pin and wire capacitance values of the ports.
<i>port</i>	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.

```
rc:/> report port -delay -driver -load ck1 e_out[6] ena
External Delays & Exceptions
-----
      Port   Dir   Clock   Rise   Fall   Ext Delay       Exception
                Delay   Delay   Object   Object/Type
-----
    ck1     in    CLK1    700.0  700.0  in_del_1           N/A
            CLK2    500.0  500.0  in_del_2
    e_out[6]  out   CLK1    200.0  200.0  ou_del_1  del_1 (path_delay)
            CLK2    300.0  300.0  ou_del_2
                    300.0  300.0  ouTrxtI
    ena     inout  CLK1    700.0  700.0  in_del_1           N/A
            CLK2    500.0  500.0  in_del_2
    ena     inout  CLK1    200.0  200.0  ou_del_1  del_1 (path_delay)
            CLK2    300.0  300.0  ou_del_2
                    300.0  300.0  ouTrxtI
```

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

#### **Related Information**

[Generating a Port Report in Setting Constraints and Performing Timing Analysis in Encounter RTL Compiler](#)

# Command Reference for Encounter RTL Compiler

## Analysis and Report

# report power

```
report power
  { -rtl_cross_reference [-detail]
    [-flat [-nworst number]] [-sort mode]
    [design | instance]...
    [-mode mode | -power_mode power_mode] [-tcf_summary]
  | [-hier | -flat [-nworst number]] [-depth number]
    [-sort mode] [design | instance | net | pin]...
    [-mode mode | -power_mode power_mode] [-tcf_summary]
  | -clock_tree [clock]...
    -width float -height float }
  [-verbose] [> file]
```

Reports the power consumed. The information returned depends on your current *position* in the design hierarchy and on the specified objects. If no objects are specified, the report is given for the design or instance at the current *position* in the design hierarchy.

With the `-rtl_cross_reference` option specified, the power consumed by the instances is cross-referenced to the corresponding line in the RTL files. Set the `hdl_track_filename row_col` attribute to true before elaboration to enable filename, column, and line number tracking.

**Note:** Nets connected to primary inputs and outputs are only reported at the top-level of an instance-based power report.

## Options and Arguments

<i>clock</i>	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.
<i>-clock_tree</i>	Estimates the power of the clock tree.  You can use this option with generic and mapped netlists.
<i>-depth number</i>	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.  <b>Note:</b> This option applies to instance-based power reports, but is not supported with the <i>-rtl_cross_reference</i> option.  <b>Default:</b> infinite (all levels of the hierarchy)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

<i>design</i>	Specifies the design for which you want the power to be reported. Specify the path name to the design.  If your current <i>position</i> 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.
<i>-detail</i>	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.  <b>Note:</b> This option only applies if you specified the <i>-rtl_cross_reference</i> option.
<i>file</i>	Specifies the name of the file to which to write the report.
<i>-flat</i>	Reports the power of leaf instances (combinational and sequential instances) starting from the <i>current</i> position in the hierarchy. <ul style="list-style-type: none"><li>■ If you perform a gate-level power analysis, the number of hierarchical levels that are expanded depends on the setting of the <i>-depth</i> option.</li><li>■ If you perform RTL power analysis using the <i>-rtl_cross_reference</i> option, power information for all modules in the current hierarchy is shown.</li></ul> <b>Note:</b> This option only applies to instance-based power reports.
<i>-full_instance_names</i>	Reports the full path names of the instances.
<i>-height float</i>	Specifies the estimated chip height (in microns).  <b>Note:</b> This option can only be specified with the <i>-clock_tree</i> option and is optional if a DEF file was read in.
<i>-hier</i>	Reports the power of hierarchical instances. The number of hierarchical levels shown depends on the setting of the <i>-depth</i> option.  If you specify neither the <i>-flat</i> or <i>-hier</i> option, the RC-LP engine uses the <i>-hier</i> option by default.  <b>Note:</b> This option applies to instance-based power reports, but is not supported with the <i>-rtl_cross_reference</i> option.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

<i>instance</i>	Specifies the instance for which you want the power to be reported. Specify the path name to the instance.		
<i>-mode mode</i>	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.		
<i>net</i>	Specifies the net for which you want the power to be reported. Specify the path name to the net.  <b>Note:</b> Net-based power reports are not supported with the <i>-rtl_cross_reference</i> option.		
<i>-nworst number</i>	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 <i>-flat</i> option.		
<i>pin</i>	Specifies the pin for which you want the power to be reported. Specify the path name to the pin.		
<i>-power_mode power_mode</i>	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 <i>create_power_mode</i> 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.  <b>Note:</b> This option cannot be used with the <i>-clock_tree</i> option.		
<i>-rtl_cross_reference</i>	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.		
<i>-sort mode</i>	Indicates how to sort the report.  The following modes are available for <i>instance</i> -based reports:  <table><tr><td><i>dynamic</i></td><td>Sorts by descending total dynamic power, which is the sum of the internal and net power.</td></tr></table>	<i>dynamic</i>	Sorts by descending total dynamic power, which is the sum of the internal and net power.
<i>dynamic</i>	Sorts by descending total dynamic power, which is the sum of the internal and net power.		

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

	<b>Note:</b> If you specified the <code>-rtl_cross_reference</code> option, this mode is available without any other options or with the <code>-verbose</code> option.
file	Sorts by RTL file and line number.
internal	<b>Note:</b> This option applies only to RTL power analysis and is the default for RTL power analysis.
leakage	Sorts by descending internal power.
net	<b>Note:</b> If you specified the <code>-rtl_cross_reference</code> option, this mode is only available if you also specified the <code>-detail</code> or <code>-verbose</code> option.
total	Sorts by descending leakage power.
	Sorts by descending net power.
	<b>Note:</b> If you specified the <code>-rtl_cross_reference</code> option, this mode is only available if you also specified the <code>-detail</code> or <code>-verbose</code> option.
dynamic (default)	Sorts by descending total dynamic power.
load	Sorts by descending capacitive load on the net.
net	Sorts by descending total switching power.
prob	Sorts by descending probability of the nets being high.
rate	Sorts by descending toggle rates of the nets.

The following modes are available for *net*-based reports:

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

**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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### Examples

- The following command requests a basic RTL power analysis of design mult\_bit\_muxed\_add.

```
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          Leakage   Dynamic   Total
      Power(nW)  Power(nW)  Power(nW)
-----
mult_bit_muxed_add    71.146  1683.918 1755.064

File           Row   Leakage   Dynamic   Total
      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.

```
rc:/> report power -rtl -flat
=====
...
Technology library:    xxx yy
Operating conditions: _nominal_ (balanced_tree)
Wireload mode:         enclosed
=====

Design          Leakage   Dynamic   Total
      Power(nW)  Power(nW)  Power(nW)
-----
mult_bit_muxed_add    71.146  1683.918 1755.064

File           Row   Leakage   Dynamic   Total
      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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following command requests a detailed RTL power analysis report.

```
rc:/> report power -rtl -detail -flat
=====
...
Module:          mult_bit_muxed_add
...
=====
Design          Leakage   Internal   Net
              Power(nW) Power(nW) Power(nW)
-----
mult_bit_muxed_add      71.146  1148.102  535.815
-----
Primary Input          Leakage   Internal   Net
              Power(nW) Power(nW) Power(nW)
-----
a[1]                  0.000    0.000    34.050
a[0]                  0.000    0.000    34.050
b[1]                  0.000    0.000    34.050
b[0]                  0.000    0.000    34.050
c[1]                  0.000    0.000    34.050
c[0]                  0.000    0.000    34.050
d[1]                  0.000    0.000    34.050
d[0]                  0.000    0.000    34.050
s                      0.000    0.000    52.536
-----
File        Row     RTL Line Instances  Leakage   Internal   Net
              Power(nW) Power(nW) Power(nW)
-----
muxed_add.v    8      {if (s) begin} mux..8_5_g1      14.199  469.091  82.695
                mux_y_8_5_g1
-----
muxed_add.v    9      {y = a + c;}           g2       28.473  334.117  62.021
                g2
-----
muxed_add.v   12      {y = b + d;}           g2       28.473  344.894  66.156
                g2
-----
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following command shows the clock tree power estimation for clock `clk`.

```
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
```

- The following command reports the power (after synthesis) at the current level in the hierarchy, which is the design. With the `-hier` option specified, the report lists the design and its hierarchical instances.

```
rc:/designs/mult_bit_muxed_add> report power -hier
=====
...
=====

              Leakage      Dynamic      Total
Instance      Cells   Power (nW)  Power (nW)  Power (nW)
-----
mult_bit_muxed_add      6      71.146   1683.862   1755.008
  ma0                   3      35.573   724.869    760.442
  ma1                   3      35.573   643.092    678.665
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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
          (Int+Net)    (Leak+Dyn)
Instance Cells 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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following command reports the power for an instance and sorts the report according to descending net power.

```
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.

```
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

      Net        Net        Toggle
      (asserted *) Power (nW) Prob. Rate (/ns) Cap. (nF)
-----
a[1]            27.945  0.500       0.021     2.300
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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
=====
```

Instance	Domain (Voltage)	Leakage Cells Power(nW)	Dynamic Power(nW)	Total Power(nW)
counter	PDcore(0.81v)	142    7.076	9192.022	9199.099
monitor_power	PDmon(0.81v)	84     3.720	4050.236	4053.956
adder_counter	PDadd(0.81v)	25     1.215	1573.961	1575.175
bcd_counter	PDbcd(0.81v)	12     1.088	1697.572	1698.659
binary_counter	PDbin(0.81v)	18     1.012	1565.877	1566.888

```
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.
```

Instance	Library Domain	Leakage Cells Power(nW)	Dynamic Power(nW)	Total Power(nW)
counter	umc_1v	142    11.023	33056.713	33067.735
monitor_power	umc_1v	84     5.805	14774.642	14780.447
adder_counter	umc_1v	25     1.884	4185.202	4187.086
bcd_counter	umc_1v	12     1.697	6138.779	6140.477
binary_counter	umc_1v	18     1.569	6542.145	6543.714

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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 969  
    [synthesize](#) on page 379

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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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 (*)      -                  true           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
-----
5
```

- 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 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      0.8      OFF      OFF
PD4            0.8      0.8      0.8      OFF
-----
5
```

## Related Information

Affected by these commands: [read\\_power\\_intent](#) on page 1030

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### **report proto**

```
report proto [-hdl] [-feasible_targets [-num integer]] [> file]
```

Reports information about the prototype synthesis flow.



Limited access feature.

#### **Options and Arguments**

<i>file</i>	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 <i>integer</i>	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.

```
rc:> report proto -feasible_targets
=====
...
=====

=====
No. Endpoint          Path Adjust      Path Adjust
           Value(ps)    Exception
=====
1   top/piano/u_reg/d    1500        proto_ft_adj_1
2   top/piano/p_reg/d    1500        proto_ft_adj_2
3   top/piano/p_reg/d    1500        proto_ft_adj_2
4   top/q_reg1/d         450         proto_ft_zipped_1
5   top/q_reg2/d         450         proto_ft_zipped_2
=====
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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

#### [Prototype Synthesis Flow in Encounter RTL Compiler Synthesis Flows](#)

Affected by these commands:      [elaborate](#)

[synthesize](#)

Related attributes:      [proto\\_feasible\\_target](#)

[proto\\_hdl](#)

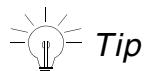
# Command Reference for Encounter RTL Compiler

## Analysis and Report

## 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.



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.

# Command Reference for Encounter RTL Compiler

## Analysis and Report

### 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:                   rc0pt55
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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 (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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following command requests to report the number of gates on the critical path per cost group data. This adds a No of gates on Critical Path column the table under Timing. The rest of the report does not change.

```
rc:\> report qor -levels_of_logic
=====
...
=====
```

#### Timing

Cost Group	Critical Path	Slack	TNS	No of gates on Critical Path	Violating Paths
C2C	No paths	0		0	0
C2O	2015.1	0		0	0
CLK1	2019.9	0		1	0
default	No paths	0		0	0
I2C	1707.2	0		2	0
I2O	No paths	0		0	0
Total		0		0	0

- The following example reports the QoR data for a Dynamic Voltage Frequency Scaling (DVFS) design (design with multiple power modes). In this case, an additional Mode column is added to the table under Timing. The rest of the report does not change.

#### Timing

Mode	Cost Group	Critical Path	Slack	Violating Paths
m1	default	No paths		
	I2C		-202.6	1
	C2O		-116.5	1
	C2C	No paths		
	I2O	No paths		
m2	default		-202.6	2
	I2C	No paths		
	C2O	No paths		
	C2C	No paths		
...				

- 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**

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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
...
Cost      Critical           Violating
Group     Path Slack        TNS       Paths
-----
default   No paths          0
vclk1    -4944.1           -12686    38
vclk01   No paths          0
...
-----
Total                 -12686           38

Instance Count
-----
Leaf Instance Count      5977
Sequential Instance Count 546
Combinational Instance Count 5431
Hierarchical Instance Count 26

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%
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- 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             rcpt55
```

### Related Information

Affected by these attributes:      [max\\_dynamic\\_power](#)  
[max\\_leakage\\_power](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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' .....

```
Results of analyze_scan_compressibility
-----
Design      - test
Decompressor - broadcast
Compressor   - missr
mask        - widel

Achieved compression table with fullscan topup vectors
-----
IC    TATR    TDVR    ATCov.   CL      Pat-comp  Pat-fs   Cycles   Runtime   Gatecount   Area
-----  
fs    1.0     1.0     100.00%  63      -          3         336       00:00:00  -          -  
2     0.3     0.5     99.91%   32      9          5         1258      00:00:01  437       24774.3999999

Achieved compression table without fullscan topup vectors
-----
IC    TATR    TDVR    ATCov.   CL      Pat-comp  Pat-fs   Cycles
-----  
fs    1.0     1.0     100.00%  63      3          336  
2     0.4     1.2     98.76%   32      9          -          838

Total atpg runtime for exp. 00:00:01 hrs.

IC      - Inserted compression
TATR    - Test application time reduction
TDVR   - Test data volume reduction
Cov.    - Atpg coverage
CL     - Channel Length
Pat-comp - No. of compression test patterns
Pat-fs   - No. of fullscan test patterns
Runtime  - Atpg runtime
fs      - fullscan run
```

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

#### **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

<i>file</i>	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.  <b>Note:</b> If the <code>delete_unloaded_seqs</code> 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.  <b>Note:</b> This option cannot be combined with any other option.
-hier	Reports all the flops in the design.
-instance_hier <i>instance</i>	Specifies the hierarchical instance name to report flops.
<i>subdesign</i>   <i>design</i>	Specifies the design or subdesign name to report flops.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

## 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
=====
                                         Instantiated/
Register      File  Row Column  Inferred           Type
-----
sync_rst_reg  all.v 12   16    inferred  flip-flop synchronous reset
async_rst_reg all.v 21   27    inferred  flip-flop asynchronous reset
sync_set_reg  all.v 30   37    inferred  flip-flop asynchronous set
no_rst_reg    all.v 39   45    inferred  flip-flop
sync_preset_reg all.v 44   58    inferred  flip-flop synchronous set
q_reg         all.v 6    12    inferred  latch
```

- The following example reports all the sequential elements in the design.

```
rc:/> report sequential -hier
report sequential: prints a sequential instance report.
=====
...
=====
                                         Instantiated/
Register      File  Row Column  Inferred           Type
-----
m2/m3/m4/m5/o_m5_0_reg_1 hier.v 25   22  instantiated  flip-flop synchronous
                                         enable
m2/m3/m4/m5/o_m5_0_reg_2 hier.v 27   22  instantiated  flip-flop synchronous
                                         enable
m2/m3/m4/m5/o_m5_1_reg_0 hier.v 30   22  instantiated  flip-flop synchronous
                                         enable
m2/m3/m4/m5/o_m5_0_reg_0 hier.v 23   22  instantiated  flip-flop synchronous
                                         enable
.....
.....
m2/o_m2_clk1_0_reg_2      hier.v 174  27  instantiated  flip-flop synchronous
                                         enable
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following example reports a timing model.

```
rc:/> report sequential
=====
...
=====
          Instantiated/
Register      File      Row Column   Inferred      Type
-----
clockgate/g2clatch_tlat timing_model.v  22   29   instantiated  timing_model
```

- The following example reports the sequential elements that were removed during optimization.

```
rc:/> report sequential -deleted_seqs
=====
...
=====
      Reason      Instance Name
-----
merged      ctl_wr_reg[36] merged with ctl_wr_reg[0]
merged      ctl_wr_reg[46] merged with ctl_wr_reg[10]
merged      ctl_wr_reg[47] merged with ctl_wr_reg[11]
merged      ctl_wr_reg[40] merged with ctl_wr_reg[12]
merged      ctl_wr_reg[41] merged with ctl_wr_reg[13]
...
...
merged      ctl_wr_reg[3] merged with ctl_wr_reg[39]
merged      ctl_wr_reg[8] merged with ctl_wr_reg[44]
merged      ctl_wr_reg[9] merged with ctl_wr_reg[45]
unloaded    ctl_wr_reg[32]
unloaded    ctl_wr_reg[12]
unloaded    ctl_wr_reg[20]
unloaded    ctl_wr_reg[28]
```

If the `delete_unloaded_seqs` root attribute is set to false, the report may look like:

```
rc:/> report sequential -deleted_seqs
=====
...
=====
      Reason      Instance Name
-----
merged (*)  ctl_wr_reg[ 36 ] merged with ctl_wr_reg[ 0 ]
merged (*)  ctl_wr_reg[ 46 ] merged with ctl_wr_reg[ 10 ]
merged (*)  ctl_wr_reg[ 47 ] merged with ctl_wr_reg[ 11 ]
merged (*)  ctl_wr_reg[ 40 ] merged with ctl_wr_reg[ 12 ]
merged (*)  ctl_wr_reg[ 41 ] merged with ctl_wr_reg[ 13 ]
constant 0 (*) ctl_wr_reg[ 52 ]
constant 1 (*) ctl_wr_reg[ 53 ]
```

(\*) indicates that the instance is optimized but not deleted  
because root attribute '`delete_unloaded_seq`' is set to false

## **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**

<i>file</i>	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.
<i>pin</i>	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

<code>-all</code>	Reports all timing and DRC violations in the design.
<code>design</code>	Specifies the design for which you want to generate a report. By default, a report is created for all designs currently loaded in memory.
<code>file</code>	Specifies the name of the file to which to write the report.
<code>-mode mode</code>	Specifies the mode for which the report must be specified. <b>Note:</b> This option is only required for a design using a dynamic voltage frequency scaling methodology.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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 379  
                                      [create\\_mode](#) on page 317

## **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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

**-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**.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

<code>-tcf file</code>	Specifies the TCF file from which to read the switching activities.
<code>-upper_bound</code>	Runs ATPG with random fill to provide an upper bound on the estimated power.  If you specified neither the <code>-lower_bound</code> nor the <code>-upper_bound</code> option, the <code>-upper_bound</code> option will be used by default.
	<b>Note:</b> This option must be specified with the <code>-atpg</code> option.

### 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%
=====
Generated by:           version
Generated on:          date
Module:                cpu
Technology library:    typical 1.3
Operating conditions: typical (balanced_tree)
Wireload mode:         segmented
Area mode:             timing library
=====

      Leakage   Internal     Net     Dynamic      Total
                           (Internal+Net) (Leakage+Dynamic)
Instance Cells Power(nW) Power(nW) Power(nW) Power(nW) Power(nW)
-----cpu       1398     0.417  66742.159 120719.340 187461.499     187461.916
```

- The following command requests to compute the upper bounds of the test power in scan shift mode from the ATPG scan patterns with a test clock frequency of 20 MHz. This command requests to use the FULLSCAN scan structure (default).

```
report test_power -atpg -clock 20 -upper_bound -directory rc_rtp_UB \
-library $simLibs
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following command requests to compute the lower bounds of the test power in scan shift mode from the ATPG scan patterns with a test clock frequency of 20 MHZ. This command requests to use the FULLSCAN scan structure (default).

```
report test_power -atpg -clock 20 -lower_bound -directory rc_rtp_LB \
-library $simLibs
```

- The following command requests to compute the test power in capture mode from the ATPG scan patterns with a test clock frequency of 20 MHZ.

```
report test_power -atpg -clock 20 -capture -directory rc_rtp_CAP \
-library $simLibs
```

- The following command requests to compute the test power from the specified TCF file.

```
report test_power -tcf top.tcf
```

### Related Information

[Analyzing the Test Power in Design for Test in Encounter RTL Compiler](#)

Affected by this constraint:

[define\\_dft test\\_clock](#) on page 761

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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.



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.

Footnote	Meaning
(*)	Zero Slack Borrow Limit = Maximum amount that can be borrowed without violating timing on the lending side
(@)	Annotated capacitance
(a)	Net has asynchronous load pins which are being considered ideal
(b)	Timing paths are broken
(C)	Cell belongs to congested region (applies only to physical flow)
(H)	Estimated wire delay based on physical-aware mapping using virtual buffering as needed.
(i)	Net is ideal
(m)	Attribute <code>cell_delay_multiplier</code> is modified for this library cell
(P)	Instance is preserved
(p)	Instance is preserved but may be resized
(u)	Net has unmapped pin(s)
(V)	Net with virtual buffer(s)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### Options and Arguments

<code>-cost_group cost_group</code>	Reports only paths for the specified cost groups.
<code>file</code>	Specifies the name of the file to which to write the report.
<code>-encounter</code>	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.
<code>-endpoints</code>	Reports the slack at all timing endpoints in the design instead of the detailed path report. The most critical endpoints are listed first.
<code>-exceptions</code>	Reports only paths to which one of the specified exceptions applies. <b>Note:</b> This option can be combined with <code>-from</code> , <code>-through</code> , <code>-to</code> , and <code>-endpoints</code> options to further restrict the path reporting.
<code>-from {instance  external_delay  clock   port   pin}</code>	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.
<code>-full_pin_names</code>	Prints the full hierarchical path of each pin in the report. You can use these pin names from the report to paste into other commands.
<code>-gtd</code>	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.
<code>-gui</code>	Allows you to create a detailed timing report in the GUI without having to use the <i>menu</i> 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. <b>Note:</b> This option has only effect when you run the tool in GUI mode.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

-gui_phys	Opens a new timing report dialog and a new Physical Viewer that are linked.
-id <i>integer</i>	In case of multiple Physical Viewers, specifies the ID of the Physical Viewer to which to apply the command.  <b>Note:</b> 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 <i>mode</i>	Reports the worst timing across all modes or analyzes timing in one particular mode.
-num_paths <i>integer</i>	Specifies the maximum number of paths to report.  <i>Default:</i> the value of -worst  <b>Note:</b> When combined with the -endpoints option, the number of endpoints is limited to the specified number.
-paths <i>string</i>	Reports only the specified timing restricted paths. Create the string argument using the <i>specify_paths</i> command.
-physical	Reports physical information, like the x, y location. When the <i>report timing show wire length</i> root attribute is enabled, the wire length information will be shown as well.
-slack_limit <i>integer</i>	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 { <i>instance</i>   <i>port</i>   <i>pin</i> }	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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

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 example reports the slack at the four most-constrained endpoints.

```
rc:/designs/alu> report timing -endpoints -num_paths 4
=====
...
=====
Slack      Endpoint      Cost Group
-----
+30ps  aluout_reg[7]/D I2C_cg
+289ps aluout_reg[6]/D I2C_cg
+292ps aluout_reg[5]/D I2C_cg
+536ps aluout_reg[4]/D I2C_cg
```

- The following command reports only the worst path being launched by clock `clk1`.

```
rc:/> report timing -paths [specify_paths -from clk1]
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following example reports the path from input port accum[1] that has the least slack.

```
rc:/> report timing -from [find / -port accum[1]]  
=====  
...  
=====  


| Pin           | Type                               | Fanout | Load<br>(fF) | Slew<br>(ps) | Delay<br>(ps) | Arrival<br>(ps) |
|---------------|------------------------------------|--------|--------------|--------------|---------------|-----------------|
| (clock clock) | capture<br>adjustments             |        |              |              | -500          | 3000 R          |
| 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                  |        |              |              |               |                 |


```
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
```


```

- 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.

```
rc:/designs/alu> report timing -num_paths 4  
=====  
...  
=====  


path 1:



| Pin           | Type                               | Fanout | Load<br>(fF) | Slew<br>(ps) | Delay<br>(ps) | Arrival<br>(ps) |
|---------------|------------------------------------|--------|--------------|--------------|---------------|-----------------|
| (clock clock) | capture<br>adjustments             |        |              |              | -500          | 3000 R          |
| 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<br>(fF) | Slew<br>(ps) | Delay<br>(ps) | Arrival<br>(ps) |
|---------------|------------------------------------|--------|--------------|--------------|---------------|-----------------|
| (clock clock) | capture<br>adjustments             |        |              |              | -500          | 3000 R          |
| 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                  |        |              |              |               |                 |


```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

```
path 3:
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(ps)	(ps)	(ps)
...						
Exception	: 'path_adjusts/adj_1' path adjust				-500ps	
Cost Group	: 'I2C_cg' (path_group 'I2C_pg')					
Timing slack	: 292ps					
Start-point	: accum[1]					
End-point	: aluout_reg[5]/D					

```
path 4:
```

...						
Exception	: 'path_adjusts/adj_1' path adjust				-500ps	
Cost Group	: 'I2C_cg' (path_group 'I2C_pg')					
Timing slack	: 536ps					
Start-point	: accum[1]					
End-point	: aluout_reg[4]/D					

- 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)   | (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)   | (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                  |        |      |      |        |         |


```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

```
path 3:
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(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				

```
...
```

```
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)	(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				

```
...
```

```
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
```

```
=====
Generated by:          RTL Compiler (RC) version
...
=====
Timing slack :      30ps
Start-point :  accum[1]
End-point   :  aluout_reg[7]/D
```

- 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.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- 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'.
=====
.....
=====
Pin          Type        Fanout Load Slew   User   Delay Arrival
                           (fF)   (ps)   (ps)  Derate (ps)   (ps)
-----
(clock clk)  launch
inst1/CK      0
inst1/Q       fflopd    (m)     1 20.4   16 3.000 +443   443 R
inst2/A       +0
inst2/Y       443
inst6/A       528
inst6/Y       528
inst3/D       buf1      1 20.4   16 1.000 +134   662 R
inst3/CKN     fflopd_ckn 1.000 +0       662
----- 
(clock clk)  capture
                           50 F
-----
Cost Group   : 'clk' (path_group 'clk')
Timing slack : -712ps (TIMING VIOLATION)
Start-point  : inst1/CK
End-point    : inst3/D
(m) : Attribute cell_delay_multiplier is modified for this library cell.
```

- 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. .....
/designs/test/instances_hier/sub/instances_seq/f0/pins_in/CK
-----
Inputs without external driver/transition
The following primary inputs have no external driver or input transition set.
As a result the transition on the ports will be assumed as zero. .....
/designs/test/ports_in/in[0]
/designs/test/ports_in/in[1]
/designs/test/ports_in/in[2]
-----
```

# Command Reference for Encounter RTL Compiler

## Analysis and Report

Outputs without external load

The following primary outputs have no external load set. As a result the load on the ports will be assumed as zero. ....

```
/designs/test/ports_out/out
```

### Lint summary

Unconnected/logic driven clocks	0
Sequential data pins driven by a clock signal	0
Sequential clock pins without clock waveform	0
Sequential clock pins with multiple clock waveforms	1
Generated clocks without clock waveform	0
Generated clocks with incompatible options	0
Generated clocks with multi-master clock	0
Paths constrained with different clocks	0
Loop-breaking cells for combinational feedback	0
Nets with multiple drivers	0
Timing exceptions with no effect	0
Suspicious multi_cycle exceptions	0
Pins/ports with conflicting case constants	0
Inputs without clocked external delays	0
Outputs without clocked external delays	0
Inputs without external driver/transition	3
Outputs without external load	1
Exceptions with invalid timing start-/endpoints	0

Total: 5

- The following example reports the physical information (if the design has been placed):

```
rc:/> report timing -physical
=====
...
=====
Pin           Type     Fanout Load Slew Delay Arrival  Location
                (fF)   (ps)   (ps)   (ps)   (x,      y)
=====
(clock clock1)    launch          0 R
wr_addr_reg[0]/CK (@) SDFFRHQX1    2  6.0  118 +311   311 R  (107640, 51660)
g154/A          (@) CLKMX2X2     3 10.9  105 +206   517 R  (102580, 51660)
g146/B          (@) ADDHXL       1  4.6  167 +189   706 R  (101660, 59040)
g145/A          (@)             167 +0     706
...
----- (clock clock1)    capture        10000 R
-----
Timing slack : 7997ps
Start-point : wr_addr_reg[0]/CK
End-point   : wr_addr_reg[7]/D
(@) : Annotated capacitance.
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

- The following report shows the timing report with the **-encounter** option.

```
rc:/> report timing -encounter
```

```
=====
.....
=====
Path 1: MET Setup Check with Pin ff2/CK
Endpoint: ff2/D (v) checked with leading edge of 'CK'
Beginpoint: ff1/Q (^) triggered by leading edge of 'CK'
Other End Arrival Time 12000
- Setup 304
- Uncertainty 1000
= Required Time 10696
- Arrival Time 2373
= Slack Time 8323
    Clock Rise Edge 0
    + Clock Network Latency (Ideal) 2000
    = Beginpoint Arrival Time 2000
-----
      Pin   Edge   Cell   Fanout Load Slew Delay Arrival Required
                  (ff)   (ps)   (ps)  Time(ps) Time(ps)
-----
      ff1/CK   ^          0        2000 10323
      ff1/Q    ^     DFFHQX1   1   7.1  160  +310  2310 10633
      b2/A     ^          +0        2310 10633
      b2/Y    v     INVX1    1   5.7   88  +63   2373 10696
      ff2/D   <<< v   DFFHQX1          +0   2373 10696
-----
Cost Group : 'CK' (path_group 'CK')
```

- The following 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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### 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 317

[specify\\_paths](#) on page 353

[synthesize](#) on page 379

[dc::set\\_timing\\_derate](#)

Affected by this attribute:      [cell\\_delay\\_multiplier](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<i>object</i>	Specifies the library for which you want to generate a report.
<i>file</i>	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](#)

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### **report utilization**

```
report utilization [>file]
```

Reports the floorplan utilization for the design.

#### **Options and Arguments**

<i>file</i>	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
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

#### **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**

<code>-depth <i>integer</i></code>	Specifies the number of levels of recursion.
<code>-min_count <i>integer</i></code>	Specifies the minimum instance count per block.
<code><i>file</i></code>	Redirects the report to the specified file.

#### **Example**

- The following example shows the defect-limited yield impact for library cell defects.

```
rc:/> report yield
```

Instance	Cells	Cell Area	Cost	Yield %
cpu	470	659	1.600606e-05	99.9984
alu1	248	283	7.606708e-06	99.9992
pcount1	65	92	2.215669e-06	99.9998
ireg1	33	88	1.629471e-06	99.9998
accum1	33	88	1.629471e-06	99.9998
decode1	50	67	1.568901e-06	99.9998

#### **Related Information**

##### [Design For Manufacturing Flow in Encounter RTL Compiler Synthesis Flows](#)

Affected by this command:	<a href="#"><u>read_dfm</u></a>
Related command:	<a href="#"><u>report_gates -yield</u></a>
Affected by this attribute:	<a href="#"><u>optimize_yield</u></a>

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## 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

<code>add_metric</code>	Adds a user-defined metric to be tracked.
<code>log</code>	Computes the metrics at the stage where it is called and stores the data with the specified tag name.
<code>read</code>	Reads the metrics from the specified file into RTL Compiler.
<code>remove_metric</code>	Removes a user-defined metric.
<code>report</code>	Reports the metrics that were recorded at various stages or that were read in.
<code>reset</code>	Resets the metrics recorded or read in during the session.
<code>run_stage_ids</code>	Reports a summary of run IDs and the associated stage IDs and run description in the RC session.
<code>write</code>	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:	<a href="#"><u>statistics add_metric</u></a> on page 557
	<a href="#"><u>statistics log</u></a> on page 558
	<a href="#"><u>statistics read</u></a> on page 560
	<a href="#"><u>statistics remove_metric</u></a> on page 561
	<a href="#"><u>statistics report</u></a> on page 562
	<a href="#"><u>statistics reset</u></a> on page 565

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

[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 <i>metric</i>	Specifies the name of the user-defined metric.
-function <i>function</i>	Specifies the Tcl function to execute to compute the metric.
<i>argument</i>	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
```

## **Command Reference for Encounter RTL Compiler**

### Analysis and Report

---

#### **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

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## **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**

<code>-file file</code>	Specifies the file to read the statistics information from.
<code>-reset</code>	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:	<u><a href="#">statistics run_stage_ids</a></u> on page 566
Related command:	<u><a href="#">statistics write</a></u> on page 567
Related attributes:	<u><a href="#">statistics run_id</a></u> <u><a href="#">statistics run_description</a></u>

## **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**

`-compare run_id`      Specifies the run that you want to compare to the run specified with the `-run_id` option.

`-compare_stage_id stage_tag`      Specifies the stage(s) of the run specified with the `-compare` option that you want to list in the report.

**Note:** You cannot specify this option without the `-stage_id` option.

`-ignore_user_defined`      Specifies to ignore the user-defined metrics in the report.

`-run_id run_id`      Specifies the run for which you want to see the statistics information.

`-stage_id string`      Specifies the stage(s) of the run specified with the `-run_id` option that you want to list in the report.  
If you omit this option, the report will show statistics information for all stages.

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

## Examples

- The following command shows the report for run sample1 for the incremental stage.

```
statistics report -run_id sample1 -stage_id incremental
QOR statistics summary
-----
Metric          incremental
-----
WNS.I2O          no_value
WNS.I2C          1387.9
WNS.CLK1         2219.9
WNS.C2C          no_value
WNS.C2O          2187.2
WNS.default      no_value
TNS              0
Violating_paths  0
runtime          0.14
memory           0.00
Leakage_power    327.19
Net_power        362656.25
Internal_power   19369.41
Clock_gating_instances 0
total_net_length n/a
average_net_length n/a
routing_congestion n/a
utilization      0.0
Inverter_count   10
Buffer_count     0
timing_model_count 0
sequential_count 16
unresolved_count 0
logic_count      13
Total_area       538.01
Cell_area        538.01
Net_area         0.00
```

- The following report includes header and footer information defined with the `statistics add_metric -header` and `-footer` options.

```
QOR statistics summary
-----
Machine_Info rcae010 Intel(R) Xeon(TM) CPU 2.80GHz 2793.238Mhz 32bits 2_cores
Date Aug17-21:56:48
...
Metric          elaborate generic global_map incremental
-----
WNS.I2O          n/a      no_value  no_value  no_value
WNS.I2C          n/a      1533.5   1912.6   1387.9
WNS.CLK1         n/a      no_value  2219.9   2219.9
WNS.C2C          n/a      no_value  no_value  no_value
WNS.C2O          n/a      2187.2   2314.1   2187.2
WNS.default      no_value no_value  no_value  no_value
TNS              0         0         0         0
Violating_paths  0         0         0         0
...
Elapsed_Time 266
Super_Thread_Total_Runtime 245.000
```

## Command Reference for Encounter RTL Compiler

### Analysis and Report

---

- The following command does a comparison between two runs.

```
statistics report -run_id medium_effort -compare high_effort -stage global_map  
QoR statistics summary  
-----
```

Metric	global_map.medium_effort	global_map.high_effort	%diff
WNS.I2O	no_value	no_value	n/a
WNS.I2C	1912.6	1887.6	1.31
WNS.CLK1	2219.9	2219.9	0.0
WNS.C2C	no_value	no_value	n/a
WNS.C2O	2314.1	2314.1	0.0
WNS.default	no_value	no_value	n/a
TNS	0	0	n/a
Violating_paths	0	0	n/a
runtime	1.74	0.32	81.61
memory	8.05	1.11	86.21
Leakage_power	1730.88	1730.88	0.0
Net_power	453828.12	453828.12	0.0
Internal_power	59389.35	59389.35	0.0
Clock_gating_instances	0	0	n/a
total_net_length	n/a	n/a	n/a
average_net_length	n/a	n/a	n/a
routing_congestion	n/a	n/a	n/a
utilization	0.0	0.0	n/a
Inverter_count	10	10	0.0
Buffer_count	0	0	n/a
timing_model_count	0	0	n/a
sequential_count	16	16	0.0
unresolved_count	0	0	n/a
logic_count	13	13	0.0
Total_area	995.70	995.70	0.0
Cell_area	995.70	995.70	0.0
Net_area	0.00	0.00	n/a
state	mapped	mapped	n/a

### Related Information

#### [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**

```
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
rc:/> statistics reset
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**

`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**

```
statistics write  
[-to_file file]
```

Writes out the statistics information (QoR metrics) recorded at various stages to the specified database file.

**Note:** When you specify an existing file, the tool will overwrite the existing data.

### **Options and Arguments**

**-to\_file file**      Specifies the file to which to write the statistics information.

If you omit this option, the data is recorded in the file determined by the `statistics_db_file` attribute.

### **Example**

```
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**

<i>string</i>	Specifies a user-defined string which allows you to identify at which stage in the design the information was requested.  <i>Default:</i> undefined.
<i>file</i>	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**

<i>file</i>	Specifies the name of the file to which the report must be written.  <i>Default:</i> vtim_ets_timing_rpt
<i>-include file</i>	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.
<i>-keep_temp_dir</i>	Does not remove the temporary (.vtim_ets) directory in which the tool generates the SDC file or netlist.
<i>-libs lib_list</i>	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.
<i>-netlist path</i>	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.
<i>-rep_tim_str command</i>	Specifies a string that contains a single Encounter Timing System report command.
<i>-sdc file_list</i>	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.

```
validate_timing -netlist test.v -libs $env(REGLIBS)/tutorial.lib \
    -include ets_include
```

Because no SDC file was specified, the vtim\_run\_ets.sdc file is generated in the .vtim\_ets directory.

- The following command directly specifies which Encounter Timing System command to be execute.

```
validate_timing -rep_tim_str "report_timing"
```

---

# **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\\_spf](#) 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\\_spf](#) 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.

## Command Reference for Encounter RTL Compiler

### Physical

---

**-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
```

```
=====  
Generated by: Cadence Encounter(R) RTL Compiler v14.20-p002_1  
Generated on: Oct 17 2014 15:18:10  
Module: rct  
=====  
=====  
check_floorplan SUMMARY  
=====  
-----  
Floorplan Object | Status | #Violations | #Total | ID | Violation Reason  
-----  
Core/Die Box | PASS | 0 | 1 | |  
Cell Sites | PASS | 0 | 40 | |  
Hard macros | WARN | 1 | 3 | FPLN-5 | not pre-placed and fixed  
-----
```

The checker finds : 1 category does not pass the checking.  
Please check it and make necessary corrections before proceeding with your synthesis job.

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s

■ The following command reports the areas with unusual utilization in the design.

```
rc:/> check_floorplan -utilization
```

```
=====  
...  
Module: rct  
=====  
=====  
check_floorplan SUMMARY  
=====  
-----  
Floorplan Object| Status | #Violations| #Total | ID | Violation Reason  
-----  
Core/Die Box | PASS | 0 | 1 | |  
Cell Sites | PASS | 0 | 40 | |  
Top/Regions Util| WARN | 4 | 4 | FPLN-6 | cells out of bound, over- or under-utilized  
-----
```

The checker finds : 1 category does not pass the checking.  
Please check it and make necessary corrections before proceeding with your synthesis job.

Done checking floorplan - Elapsed time 0s, CPU time: 0.00s

## Command Reference for Encounter RTL Compiler

### Physical

---

- 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
```

## Command Reference for Encounter RTL Compiler

### Physical

- 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

## Command Reference for Encounter RTL Compiler

### Physical

---

#### **check\_placement**

```
check_placement design  
[-file file] [-verbose]
```

Checks the placement legality and highlights illegal objects. The command also returns some instance status statistics.

**Note:** This command applies only to the RC-P flow.

#### **Options and Arguments**

<code>-file file</code>	Specifies the file to which to write the report.
<code>-verbose</code>	Specifies to generate a verbose report.
<code>design</code>	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...
```

## Command Reference for Encounter RTL Compiler

### Physical

---

```
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**

<code>-design <i>design</i></code>	Specifies the name of the design for which to create the group.
<code>-region <i>name</i></code>	Associates the specified region with the group.
<code><i>group</i></code>	Specifies the name of the group to be created.
<code><i>instance_list</i></code>	Specifies the instances that belong to this group.

### **Example**

The following example creates group `gr_reg` with two instances:

```
create_group mygroup -region myreg [find / -inst a_reg_3] [find / -inst a_reg_5]
```

### **Related Information**

Affects this command: [synthesize](#) on page 379

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 379

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 <i>integer</i>	Specifies the halo width at the bottom of the instance. Specify the distance in DB units.
-left <i>integer</i>	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 <code>-no_update</code> option to update the internal data structures in one operation. This will speed up the operation.
-right <i>integer</i>	Specifies the halo width at the right of the instance. Specify the distance in DB units.
-soft	Creates a <u>soft</u> placement blockage.
-top <i>integer</i>	Specifies the halo width at the top of the instance. Specify the distance in DB units.
<i>instance</i>	Specifies the instance around which the placement halo is created.

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

### **Related Information**

Affects this command:	<a href="#">synthesize</a> on page 379
Related commands:	<a href="#">create_placement_blockage</a> on page 580 <a href="#">move_blockage</a> on page 598 <a href="#">resize_blockage</a> 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:

[synthesize](#) on page 379

## **Command Reference for Encounter RTL Compiler**

### **Physical**

---

Related commands:      [move\\_region](#) on page 601

[resize\\_region](#) on page 612

Related Attributes:      [Region Attributes](#)

## Command Reference for Encounter RTL Compiler

### Physical

---

#### **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**

<code>-design <i>design</i></code>	Specifies the name of the design for which to create the region.
<code>-height <i>integer</i></code>	Specifies the height of the row.
<code>-llx <i>integer</i></code>	Specifies the lower left X coordinate of the row.
<code>-lly <i>integer</i></code>	Specifies the lower left Y coordinate of the row.
<code>-macro <i>name</i></code>	Specifies the name of the LEF site to be used for the row.
<code>-no_update</code>	Prevents an update of the GUI when the new row is created.
<code>-orientation</code>	Specifies the orientation of the row to be created.
<code>-width <i>integer</i></code>	Specifies the width of the row.
<code><i>row</i></code>	Specifies the name of the row to be created.

#### **Example**

The following command creates row `my_row` for design `DTMF`.

```
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 379

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.

## Command Reference for Encounter RTL Compiler

### Physical

---

#### Related Information

Affects this command: [synthesize](#) on page 379

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 379

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**

<code>-design <i>design</i></code>	Specifies the name of the design in which the tracks must be created.
<code>-layer <i>string</i></code>	Specifies the routing layer for which the tracks are created.
<code>-num <i>integer</i></code>	Specifies the number of tracks to be created.
<code>-start <i>integer</i></code>	Specifies the starting X or Y coordinate. X coordinate applies to vertical tracks, while the Y coordinate applies to horizontal tracks.
<code>-step <i>integer</i></code>	Specifies the spacing between the tracks.
<code>-vertical</code>	Specifies to create vertical tracks. By default, a horizontal track is created.

### **Related Information**

Related command:      [synthesize](#) on page 379

## **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**

<code>-highlight</code>	Highlights the cell movement with respect to their last stored location.
<code>-initialize</code>	Stores the location of the instances at the time the command is given with this option.
<code>-min_distance <i>string</i></code>	<p>Limits the highlighting to cells that have been moved more than the specified distance. Specify the distance in microns.</p> <p>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.</p>

### **Example**

Following illustrates the usage:

```
def_move -initialize
synthesize -to_placed -incr
def_move -highlight
```

### **Related Information**

Related command: [synthesize](#) on page 379

## **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: all.lef
# CAP-TABLE: typical.captbl
# CAP SCALE: 1.0
# RES SCALE: 1.0
# ASPECT RATIO: 0.9814
```

If the design data is not inconsistent, the following message will be issued:

```
Warning : Inconsistent data. [PHYS-600]
          : Original design "DTMF_CHIP" not found in current session.
          : Input data used to create PLE correlation file is different from data
used in this session. This might lead to invalid results. Check design data.
```



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**

<i>design</i>	Specifies the design for which to generate the PLE data.
<i>-outfile file</i>	Specifies the name of the output file in which to store the PLE correlation data.

## Command Reference for Encounter RTL Compiler

### Physical

---

#### Example

In the beginning of the synthesis process, use the following flow to generate the PLE data:

```
set_attribute library library_list /
set_attribute lef_library lef_files /
read_hdl hdl_file
elaborate
read_def def_file
generate_ple_model design -outfile ple_file
```

In subsequent sessions use the following flow to load the PLE data:

```
set_attribute library library_list /
set_attribute lef_library lef_files /
read_hdl hdl_file
elaborate
decrypt ple_file
...
```

#### Related Information

Affects these commands:      [synthesize -to\\_mapped](#)  
                                [synthesize -to\\_placed](#)

## **generate\_reports**

```
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**

<code>-encounter</code>	Specifies to use the QoS statistics from the latest Encounter run. By default, the statistics are used from RTL Compiler.
<code>-outdir <i>path</i></code>	Specifies the path to the directory where the output data should be stored.
<code>-tag <i>string</i></code>	Specifies the tag name for reports generated at this stage.

### **Example**

Assume the following `generate_reports` are executed.

```
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
```

## **Command Reference for Encounter RTL Compiler**

### **Physical**

---

#### **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.

## Command Reference for Encounter RTL Compiler

### Physical

---

#### 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**

<i>-dx integer</i>	Specifies to move the blockage over the specified distance in the x-direction.
<i>-dy integer</i>	Specifies to move the blockage over the specified distance in the y-direction.
<i>-x integer</i>	Specifies the new x-coordinate for the lower left corner of the blockage.
<i>-y integer</i>	Specifies the new y-coordinate for the lower left corner of the blockage.
<i>blockage</i>	Specifies the name of the blockage to be moved.

### **Examples**

- The following command moves blockage `my_blk` over 20 DB units in the vertical direction:

```
move_blockage -dy 20 [find . -blockage my_blk]
```

### **Related Information**

Affects this command:	<a href="#">synthesize</a> on page 379
Related commands:	<a href="#">create_placement_blockage</a> on page 580 <a href="#">create_placement_halo_blockage</a> on page 581 <a href="#">resize_blockage</a> 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**

<i>-dx integer</i>	Specifies to move the instance over the specified distance in the x-direction.
<i>-dy integer</i>	Specifies to move the instance over the specified distance in the y-direction.
<i>-x integer</i>	Specifies the new x-coordinate for the lower left corner.
<i>-y integer</i>	Specifies the new y-coordinate for the lower left corner.
<i>instance</i>	Specifies the name of the instance to be moved.

### **Examples**

- The following command moves instance PLLCLK\_INST over 2 DB units in the horizontal direction:

```
move_instance -dx 2 [find . -inst PLLCLK_INST]
```

## **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**

<i>-dx integer</i>	Specifies to move the port over the specified distance in the x-direction.
<i>-dy integer</i>	Specifies to move the port over the specified distance in the y-direction.
<i>-x integer</i>	Specifies the new x-coordinate for the center of the port.
<i>-y integer</i>	Specifies the new y-coordinate for the center of the port.
<i>port</i>	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**

<i>-dx integer</i>	Specifies to move the region over the specified distance in the x-direction.
<i>-dy integer</i>	Specifies to move the region over the specified distance in the y-direction.
<i>-x integer</i>	Specifies the new x-coordinate for the lower left corner of the region.
<i>-y integer</i>	Specifies the new y-coordinate for the lower left corner of the region.
<i>region</i>	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:	<a href="#">synthesize</a> on page 379
Related commands:	<a href="#">create_region</a> on page 582 <a href="#">resize_region</a> on page 612

#### **read\_def**

```
read_def
  [-design design]
  [-create_ports] [-power_switch_insert]
  [-no_nets] [-no_specialnets] [-no_vias]
  [-keep_filler_cells] [-keep_welltap_cells]
  [-keep_routed_nets]
  [-hierarchical [-no_force_fixed] ]
  [-incremental] [-ignore_errors]
  def_file
```

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.



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**

<code>-create_ports</code>	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.
<code>def_file</code>	Specifies the DEF file.
<code>-design <i>design</i></code>	Specifies the design to which to annotate the DEF information.
<code>-hierarchical</code>	Specifies that the DEF file is hierarchical.
<code>-ignore_errors</code>	Ignores any inconsistency errors between the LEF and DEF.
<code>-incremental</code>	Specifies that the DEF file has incremental information. Therefore only updates of the blockages, components, groups, pins, and regions are needed.

## Command Reference for Encounter RTL Compiler

### Physical

---

- 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.

# Command Reference for Encounter RTL Compiler

## Physical

---

### 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
```

## Command Reference for Encounter RTL Compiler

### Physical

---

```
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**

*configuration\_file* Specifies the configuration file to load.

### **Examples**

- Since the configuration file contains information such as technology libraries, HDL files, and constraints, the `read_encounter` command should be used at the beginning of a synthesis session. After the configuration file is loaded, you can immediately synthesize or optimize the design. The following example loads the `fast.config` configuration file, then synthesizes the design to gates.

```
rc:/> read_encounter config fast.config  
rc:/> synthesize -to_mapped  
...
```

### **Related Information**

Related command: [write\\_encounter](#) on page 265

## **read\_sdp\_file**

```
read_sdp_file -file file
    [-design design] [-preserve_size_ok]
    [-hier_path path] [-origin l1x l1y]
```

Reads a relative placement file (in .sdp format) containing the SDP definitions.

### **Options and Arguments**

-design <i>design</i>	Specifies the design in which to use the SDP information.
-file <i>file</i>	Specifies the name of the SDP input file.
-hier_path <i>path</i>	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 <i>l1x l1y</i>	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. <i>Default:</i> 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.
<i>spef_file</i>	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 <i>integer</i>	Specifies the distance over which the bottom edge of the blockage must be moved.
-left <i>integer</i>	Specifies the distance over which the left edge of the blockage must be moved.
-right <i>integer</i>	Specifies the distance over which the right edge of the blockage must be moved.
-top <i>integer</i>	Specifies the distance over which the top edge of the blockage must be moved.
<i>blockage</i>	Specifies the name of the blockage to be resized.

### **Example**

The following command extends blockage `my_blk` with 20DB units on the right side.

```
resize_blockage my_blk -right 20
```

### **Related Information**

Affects this command:	<a href="#">synthesize</a> on page 379
Related commands:	<a href="#">create_placement_blockage</a> on page 580 <a href="#">create_placement_halo_blockage</a> on page 581 <a href="#">move_blockage</a> on page 598

## **resize\_region**

```
resize_region region
    [-left integer] [-right integer]
    [-top integer] [-bottom integer]
```

Resizes the specified region. All distances must be specified in DB units, where one DB unit is 1/1000 micron. You must specify at least one option.

### **Options and Arguments**

<i>-bottom integer</i>	Specifies the distance over which the bottom edge of the region must be moved.
<i>-left integer</i>	Specifies the distance over which the left edge of the region must be moved.
<i>-right integer</i>	Specifies the distance over which the right edge of the region must be moved.
<i>-top integer</i>	Specifies the distance over which the top edge of the region must be moved.
<i>region</i>	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:	<a href="#">synthesize</a> on page 379
Related commands:	<a href="#">create_region</a> on page 582 <a href="#">move_region</a> 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**

<i>libcell_list</i>	Specifies a list of libcell types to which the padding factor applies.
<i>-padding string</i>	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 379

## **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
rc:/> specify_floorplan DTMF_CHIP -core_box {100 100 1400 1400}
Error   : The design core box must lie within the die box. [PHYS-102]
[specify_floorplan]
      : core box = {100 100 1400 1400}, die box = {0 0 1200 1400}
      : Wrong coordinates were specified for the core box.
```

- The following command specifies the coordinates of the die and the core of the design.

## **Command Reference for Encounter RTL Compiler**

### **Physical**

---

```
specify_floorplan MYCHIP -die_box {0 0 1550 1500} -core_box {100 100 1400 1400}
```

#### **Related Information**

Affects this command:      [synthesize](#) on page 379

## Command Reference for Encounter RTL Compiler

### Physical

---

#### **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**

<code>-outdir <i>path</i></code>	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
=====
Metric           initial   generic   map
=====
Slack (ps):      1958.6    1934.1    1190.1
R2R (ps):        no_value   no_value   no_value
I2R (ps):        1958.6    1934.1    1190.1
R2O (ps):        2152.6    2152.6    1987.2
I2O (ps):        no_value   no_value   no_value
CG (ps):         2100.0    2100.0    2100.0
TNS (ps):         0          0          0
R2R (ps):        no_value   no_value   no_value
I2R (ps):         0          0          0
R2O (ps):         0          0          0
I2O (ps):        no_value   no_value   no_value
CG (ps):         0          0          0
Failing Paths:   0          0          0
Area:            0          0          0
Instances:       36         28         43
Utilization (%): 0.00       0.00       0.00
Tot. Net Length (um): no_value   no_value   no_value
Avg. Net Length (um): no_value   no_value   no_value
Total Overflow H:  0          0          0
Total Overflow V:  0          0          0
Route Overflow H (%): no_value   no_value   no_value
Route Overflow V (%): no_value   no_value   no_value
=====
CPU Runtime (m:s): 00:04     00:00     00:01
Real Runtime (m:s): 00:13     01:35     01:30
```

## Command Reference for Encounter RTL Compiler

### Physical

---

```
CPU Elapsed (m:s):      00:09      00:10      00:12
Real Elapsed (m:s):     00:37      02:12      03:42
Memory (MB):            66.32      66.60      73.38
=====
```

```
=====
Flow Settings:
=====
```

```
Total Runtime (m:s): 03:25
Total Memory (MB):   73.38
Executable Version:  9.1
=====
```

### Related Information

Affected by this command: [generate\\_reports](#) on page 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**

<code>-design <i>design</i></code>	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**

<code>-design <i>design</i></code>	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**

<code>-design <i>design</i></code>	Specifies the design for which the gcell utilization values must be updated.
------------------------------------	--

## Command Reference for Encounter RTL Compiler

### Physical

---

#### **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**

<i>design</i>	Specifies a particular design for which to write out the floorplan. Only one design can be specified at a time.
<i>file</i>	Redirects the floorplan to the specified file.
<b>-hierarchical <i>instance</i></b>	Specifies the hierarchical instance for which to write out the DEF.
<b>-ignore_groups</b>	Discards the instance groups that are defined in RC. These come from the input DEF ( <b>GROUPS</b> section). The output DEF will have no <b>GROUPS</b> or <b>REGIONS</b> sections.
<b>-ignore_placed_instances</b>	Only writes preplaced instances (+ <b>FIXED</b> 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 <b>COMPONENTS</b> section.
<b>-scan_chains</b>	Specifies that scan chain information should be included in the floorplan.
<b>-version <i>string</i></b>	Specifies the DEF version to write out. <i>Default:</i> 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 <i>file</i>           | Specifies the name of the SDP output file.<br><i>Default:</i> Output is written to the screen. |
| -top_group <i>sdp_group</i> | 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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- [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 761
- [define\\_dft\\_test\\_mode](#) on page 765
- [dft\\_trace\\_back](#) on page 769
- [fix\\_dft\\_violations](#) on page 771
- [fix\\_scan\\_path\\_inversions](#) on page 775
- [identify\\_domain\\_crossing\\_pins\\_for\\_cgic\\_and\\_scan\\_abstracts](#) on page 776
- [identify\\_multibit\\_cell\\_abstract\\_scan\\_segments](#) on page 777
- [identify\\_shift\\_register\\_scan\\_segments](#) on page 779
- [identify\\_test\\_mode\\_registers](#) on page 781

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- [insert\\_dft](#) on page 784
- [insert\\_dft\\_boundary\\_scan](#) on page 787
- [insert\\_dft\\_compression\\_logic](#) on page 791
- [insert\\_dft\\_dfa\\_test\\_points](#) on page 802
- [insert\\_dft\\_jtag\\_macro](#) on page 806
- [insert\\_dft\\_lockup\\_element](#) on page 810
- [insert\\_dft\\_logic\\_bist](#) on page 811
- [insert\\_dft\\_mbist](#) on page 817
- [insert\\_dft\\_opcg](#) on page 823
- [insert\\_dft\\_pmbist](#) on page 825
- [insert\\_dft\\_ptam](#) on page 830
- [insert\\_dft\\_rrfa\\_test\\_points](#) on page 833
- [insert\\_dft\\_scan\\_power\\_gating](#) on page 840
- [insert\\_dft\\_shadow\\_logic](#) on page 843
- [insert\\_dft\\_shift\\_register\\_test\\_points](#) on page 848
- [insert\\_dft\\_test\\_point](#) on page 849
- [insert\\_dft\\_user\\_test\\_point](#) on page 855
- [insert\\_dft\\_wrapper\\_cell](#) on page 857
- [insert\\_dft\\_wrapper\\_instruction\\_register](#) on page 863
- [insert\\_dft\\_wrapper\\_mode\\_decode\\_block](#) on page 865
- [insert\\_test\\_compression](#) on page 867
- [map\\_mbist\\_cgc\\_to\\_cgic](#) on page 870
- [read\\_dft\\_abstract\\_model](#) on page 871
- [read\\_io\\_speclist](#) on page 874
- [read\\_memory\\_view](#) on page 875
- [read\\_pmbist\\_interface\\_files](#) on page 877
- [replace\\_opcg\\_scan](#) on page 878

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- [replace\\_scan](#) on page 880
- [report\\_dft\\_chains](#) on page 881
- [report\\_dft\\_clock\\_domain\\_info](#) on page 882
- [report\\_dft\\_core\\_wrapper](#) on page 883
- [report\\_dft\\_registers](#) on page 884
- [report\\_dft\\_setup](#) on page 885
- [report\\_dft\\_violations](#) on page 886
- [report\\_opcg\\_equivalents](#) on page 887
- [report\\_scan\\_compressibility](#) on page 888
- [report\\_test\\_power](#) on page 889
- [reset\\_opcg\\_equivalent](#) on page 890
- [reset\\_scan\\_equivalent](#) on page 891
- [set\\_compatible\\_test\\_clocks](#) on page 892
- [set\\_opcg\\_equivalent](#) on page 894
- [set\\_scan\\_equivalent](#) on page 896
- [update\\_scan\\_chains](#) on page 898
- [write\\_atpg](#) on page 900
- [write\\_bsdl](#) on page 903
- [write\\_compression\\_macro](#) on page 906
- [write\\_dft\\_abstract\\_model](#) on page 914
- [write\\_dft\\_rtl\\_model](#) on page 917
- [write\\_et\\_atpg](#) on page 918
- [write\\_et\\_bsv](#) on page 926
- [write\\_et\\_dfa](#) on page 930
- [write\\_et\\_lbist](#) on page 934
- [write\\_et\\_mbist](#) on page 938
- [write\\_et\\_no\\_tp\\_file](#) on page 943

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

- [write\\_et\\_rrfa](#) on page 944
- [write\\_io\\_speclist](#) on page 948
- [write\\_logic\\_bist\\_macro](#) on page 950
- [write\\_mbist\\_testbench](#) on page 954
- [write\\_pmbist\\_interface\\_files](#) on page 958
- [write\\_pmbist\\_testbench](#) on page 960
- [write\\_scandef](#) on page 964

## **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 890  
[set opcg equivalent](#) on page 894

## **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 {wide1 | wide0 | wide2}] [-serial_misr]
    [-effort {low | medium | high}]
    [-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).

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

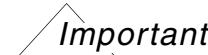
*Default:* `current_working_directory/asc`

`-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.



Neither of the preceding methods are automatically invoked by the `analyze_scan_compressibility` command or the `report_scan_compressibility` command.

`-effort {low | medium | high}`

Specifies the effort to be used for ATPG.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**Note:** Higher effort levels on larger designs cause increased runtimes.

*Default:* low

`-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 reduced 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 ..."`

# Command Reference for Encounter RTL Compiler

## Design for Test

---

**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 `widel` 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 widel \
    -ratio_list "20 50 100" -dont_run_atpg -library mylibrary -directory asc
Starting scan compressibility analysis for module 'test'

Creating work directory asc ......

Analyzing the design with the following configuration
-----
Design      - test
Decompressor - broadcast
Compressor   - xor
mask        - widel
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'.
```

# Command Reference for Encounter RTL Compiler

## Design for Test

```
.... 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 an 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'  
  
using existing work directory: './newdirnr2'  
Encounter(TM) Test product version information:  
Encounter(R) Test and Diagnostics xxxx  
  
Analyzing the design with the following configuration  
-----  
Design      - DLX_CORE  
Decompressor - broadcast  
Compressor   - xor  
mask        - wide1  
ratio       - 10 20 30  
  
Scan Registers Status  
-----  
Replacing scan  
Scan flip-flops mapped for DFT          1348      100.00%  
(Number of Scannable Registers) / (Total Register Count) =  
--> 1348 / 1348 = 100%  
  
(Number of flops mapped to scan) = 1348  
(Number of flops passing tdrc) = 1348  
  
(Number of Scannable Registers)  
(Number of flops scan mapped and passing tdrc) +  
(Number of flops in shift-register segments) +  
(Number of flops in abstract scan segments) =  
--> 1348 + 0 + 0 = 1348  
  
(Total Register Count)  
(Number of registers) +  
(Number of flops in abstract scan segments) =  
--> 1348 + 0 = 1348
```

# Command Reference for Encounter RTL Compiler

## Design for Test

(\*) The percentage of scannable registers is 100%.

Building the Fullscan Chains

Building balanced scan chains assuming all test clocks are compatible and mixing of clock edges in scan chains is allowed. If you prefer a different set of constraints, please build the scan chains before calling analyze\_scan\_compressibility.

Configuring 1 chains for 1348 scan f/f  
...chain01 of length: 1348

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

Design - DLX\_CORE  
Decompressor - broadcast  
Compressor - xor  
mask - wide1

Achieved compression table with fullscan top-off vectors

IC	TATR	TDVR	ATCov.	CL	PAT-comp	PAT-fs	Cycles	Runtime	Gatecount	Area
fs	1.0	1.0	99.99%	1348	-	305	414762	00:00:03	-	-
10	5.6	5.6	99.72%	135	301	22	73532	00:00:04	55	680.6569999999992
20	5.5	5.5	99.45%	68	324	37	75711	00:00:05	112	1357.9199999999983
30	4.4	4.4	97.83%	45	299	57	95084	00:00:13	155	1941.8260000000001

Achieved compression table without fullscan top-off vectors

IC	TATR	TDVR	ATCov.	CL	PAT-comp	PAT-fs	Cycles
fs	1.0	1.0	99.99%	1348	-	305	414762
10	9.8	9.8	99.72%	135	301	-	42212
20	17.6	17.6	99.45%	68	324	-	23586
30	27.8	27.8	97.83%	45	299	-	14946

Total ATPG runtime: 00:00:25 (hrs:min:sec).

IC - Inserted compression  
TATR - Test application time reduction  
TDVR - Test data volume reduction  
ATCov. - ATPG test mode coverage  
CL - Channel Length  
PAT-comp - Number of compression test patterns  
PAT-fs - Number of fullscan test patterns  
Runtime - ATPG runtime  
fs - Uncompressed (Fullscan) run

Completed scan compressibility analysis for module 'DLX\_CORE'

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- The following command performs MISR-based compression analysis, with `widel` masking, a ratio list of 2, sixteen scan chains.

```
rc:> analyze_scan_compressibility -library mylibrary -chains 16 \
    -compressor misr -ratio_list 2 -mask widel -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]
    [-effort {low|medium|high}]
    [-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).

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

`-effort {low | medium | high}`

Specifies the effort to be used for the ATPG-based testability analysis.

*Default:* low

`-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 765

[define\\_dft\\_test\\_clock](#) on page 761

## **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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `runet.tsv`—A template script file for Encounter Test to verify the design and its test structures

### Options and Arguments

<code>-compression</code>	Performs additional checks if the design has compression logic.
<code>design</code>	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.
<code>-directory string</code>	Specifies the working directory for Encounter Test.  <b>Note:</b> If the script files already exist, rerunning the command will overwrite them.  <i>Default:</i> <code>./check_atpg_rules</code>
<code>-library string</code>	Specifies the list of Verilog structural library files, for example, <code>file1 file2</code> . Refer to the <code>write_et_atpg_library</code> option description for additional information.  <b>Note:</b> This option is only required when you invoke this command on a mapped netlist.

### Example

```
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**

<i>design</i>	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.
<i>file</i>	Specifies the file to which to redirect the detailed output.  If no file is specified, the output is written to standard out ( <code>stdout</code> ) 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.

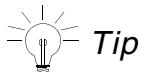
Table 11-1 lists the DFT rule violations that this command checks.

**Table 11-1 Checking For and Auto-Fixing of DFT Rule Violations**

DFT Rule Violation	Checked?	Auto-Fixed?
Uncontrollable asynchronous set or reset signals	Checked	Yes
Gated clocks and derived clocks	Checked	Yes
Flip-flop's clock port connected to tied lines	Checked	Yes
Conflicting clock and asynchronous set or reset signals	Checked	No
Tristate contention for internal and external nets	Checked(*)	Yes
Same asynchronous set or reset data race conditions	Checked(*)	Yes
Clock and data race conditions	Checked(*)	No
Floating nets violations	Checked(*)	No
X-source violations	Checked(*)	Yes
Floating conditions	Not checked	No

**Note:** (\*) indicates that the check is performed using the `-advanced` option.

To maximize fault coverage, you should try to fix any DFT rule violations, so that all flip-flops can be included in a scan chain. You can either modify the RTL or use the DFT fix capabilities using the `fix_dft_violations` command.



**Tip**  
To include the RTL file name and line number at which the DFT violation occurred in the messages produced by `check_dft_rules`, set the `hdl_track_filename_row_col root` attribute to `true` before elaboration.

You can find the objects created by the `check_dft_rules` command in:

`/designs/design/dft/test_clock_domains`

The detected violations are placed in:

`/designs/design/dft/report/violation`

## Options and Arguments

`-advanced`

Specifies to perform checking for tristate contention, same asynchronous set or reset data race conditions, clock and data race conditions, x-source generators, and floating nets violations.

`design`

Specifies the name of the top-level design to be checked. You should specify this name in case you have multiple top designs loaded.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

`-dft_configuration_mode dft_configuration_mode_name`

Specifies the object name of the scan mode to be checked. A scan mode is defined using the [define\\_dft\\_dft\\_configuration\\_mode](#) command.

Scan modes can be used to build the top-level scan chains with specific elements in different modes of operation (multi-mode), or when concatenating default scan chains into a single longer scan chain in a different scan mode of operation.

`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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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
```

# Command Reference for Encounter RTL Compiler

## Design for Test

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'.  
Info      : Added DFT object. [DFT-100]  
           : Added test clock 'clk'.  
    Checking DFT rules for async. pins  
    Checking DFT rules for shift registers.  
    Checking DFT rules for tristate nets.  
    Checking DFT rules for clock data race conditions.  
    Checking DFT rules for set reset data race conditions.  
    Checking DFT rules for x-sources.  
Detected 1 DFT rule violation(s)  
    Summary of check_dft_rules  
*****  
Number of usable scan cells: 60
```

# Command Reference for Encounter RTL Compiler

## Design for Test

---

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 761  
[define dft test mode](#) on page 765

Affects these commands:

[connect scan chains](#) on page 681  
[fix dft violations](#) on page 771  
[report dft registers](#) on page 884  
[synthesize](#) on page 379

Affected by these attributes:

[dft controllable](#)  
[dft dont scan](#)  
[dft identify test signals](#)  
[dft identify top level test clocks](#)  
[dft scan style](#)  
(instance) [preserve](#)  
(subdesign) [preserve](#)  
[dft identify xsource violations from timing models](#)

Sets these attributes:

[dft status](#)  
[dft violation](#)  
[type](#)  
[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**

---

### **MBIST Rule**

- The `Test_Control` test signal used during `insert dft_mbist` is properly controlled at the pin of all MBIST engines when the `dft_configuration_mode` is active.
  - All MBIST engine clock pins are properly controlled from design ports or internal pins for MBIST clocks defined with the `define_dft mbist_clock -internal_clock_source` command
  - All MBIST interface files are consistent with the JTAG instructions and MBIST logic inserted in the design. Portions of the design which are blackboxes containing MBIST logic are limited to interface checks.
  - The need for a user supplied mode initialization sequence by downstream tools in case MBIST direct access is implemented using internal design pins as control signals.
- 

**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_config_mode_name`

Specifies the configuration mode in which netlist tracing must be done.

*Default: mbist*

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

# Command Reference for Encounter RTL Compiler

## Design for Test

## 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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-preview** Reports the requested ratio, the maximum original scan chain length, the maximum subchain length, and the number of internal scan channels that would be created without making modifications to the netlist. Use this option to verify your compression architecture prior to inserting the compression logic.

**-target\_period integer**

Specifies a target clock period (in picoseconds) used to optimize the compression macro. If the value zero (0) is specified, synthesis is performed with a low effort compile, and without applying external (input/output delay) constraints.

*Default:* value for `test_clock` period of the scan chains being compressed

### Example

The following command requests an XOR-based compression with masking logic of type 1 and decompression logic of type XOR for design core.

```
compress_block_level_chains -channel_length 7 -decompressor xor -mask wide1 core
```

### Related Information

[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 765

Affected by these commands: [connect\\_scan\\_chains](#) on page 681

Affects these commands: [report\\_dft\\_chains](#) on page 881

[write\\_dft\\_abstract\\_model](#) on page 914

Sets these attributes: [compressed](#)

[dft\\_compression\\_signal](#)

[dft\\_mask\\_clock](#)

[dft\\_misr\\_clock](#)

[type](#)

# Command Reference for Encounter RTL Compiler

## Design for Test

---

### 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}
        [-mask_sharing_ratio integer
        [-mask_clock {port|pin} [-allow_shared_clock]] [-mask_load test_signal]
        [-mask_enable test_signal_list]
        [-create_mask_or_misr_chain
            -mask_or_misr_sdi {pin|port}
            -mask_or_misr_sdo {pin|port} [-shared_output]]]
    | -compressor hybrid
        [-misr_bypass test_signal]
        [-serial_misr_read] [-misr_observe test_signal]
        [-misr_clock {port|pin}]
        [-misr_reset_enable test_signal | -misr_reset_clock test_signal]
        [-misr_shift_enable test_signal]
        [-mask {wide0 | wide1 | wide2}]
        [-mask_sharing_ratio integer]
        [-mask_clock {port|pin}] [-mask_load test_signal]
        [-mask_enable test_signal_list]
        [-create_mask_or_misr_chain
            -mask_or_misr_sdi {pin|port}
            -mask_or_misr_sdo {pin|port} [-shared_output]]]
    | -compressor smartscan_xor
        -smartscan_ratio integer
        [-gate_shared_compression_clock] [-smartscan_no_update_stage]
        [-smartscan_serial_only | -smartscan_serial_scan_ins {pins|ports}]
        [-smartscan_parallel_access test_signal]
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

```
[-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

*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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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).

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-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 <i>shift_enable</i>	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 <i>test_mode_signal</i>	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 <i>not</i> part of the compression channels.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

*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)
...
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- The following command requests XOR-based compression logic with masking logic of type `widel`, 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 widel -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 `widel`, 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 widel \
-autocreate
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 `widel`, and requests creation of the necessary ports for the required test signals. Compared to the two second example, one extra test signal—the `mistr_reset_enable` signal—is required to reset the MISR registers, and thus five test ports are created.

```
rc:/> compress -ratio 5 -compressor misr -mask widel -auto_create
Will create a test port for 'compression_enable'
Will create a test port for 'mistr_reset_enable'.
Will create a test port for 'mistr_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 'mistr_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](#)

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### ■ Using Asymmetrical Scan Compression

Affected by these constraints:	<a href="#">define_dft_jtag_instruction</a> on page 708 <a href="#">define_dft_shift_enable</a> on page 751 <a href="#">define_dft_test_mode</a> on page 765
Affected by these commands:	<a href="#">connect_scan_chains</a> on page 681
Affects this command:	<a href="#">report_dft_chains</a> on page 881
Sets these attributes:	<a href="#">compressed</a> <a href="#">dft_compression_signal</a> <a href="#">dft_mask_clock</a> <a href="#">dft_misr_clock</a> <a href="#">type</a>

## **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\_config\_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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 765

[set\\_compatible\\_test\\_clocks](#) on page 892

Affected by this command: [check\\_dft\\_rules](#) on page 648

Affects these commands: [report\\_dft\\_chains](#) on page 881

[write\\_dft\\_abstract\\_model](#) on page 914

[write\\_scandef](#) on page 964

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 765

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*      Specifies the name of the design for which to connect the OPCG segments.

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

This option is required if multiple designs are loaded.

-chains *actual\_scan\_chains*

Specifies the names of the actual scan chains to which the OPCG segments must be prepended.

-preview

Shows the potential connections for the OPCG segments, without making any modifications to the netlist.

**-use\_ports\_for\_side\_scan\_connections port\_list**

Specifies the ports to be used as scan data input pins when making the connections for the side scan chains.

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.

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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 823

## **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-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.  <b>Note:</b> Use this option only in the physical flow after you have already placed the design using <code>synthesize -to_placed</code> . 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 906

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 892

## Command Reference for Encounter RTL Compiler

### Design for Test

---

Affected by these commands: [check\\_dft\\_rules](#) on page 648

[fix\\_dft\\_violations](#) on page 771

[synthesize](#) on page 379

Affects this command: [concat\\_scan\\_chains](#) on page 675

[report\\_dft\\_chains](#) on page 881

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

opcg_trigger	Defines the trigger signal that will enable the trigger macro associated with the specified oscillator source.
osc_source	Defines an oscillator source on the output pin of a PLL instance that will drive the OPCG logic.
pmbist_direct_access	Defines programmable MBIST direct access interface pins or ports which can be used as an alternative access mechanism for PMBIST.
preserved_segment	Defines a preserved scan-chain segment object.
scan_chain	Defines a scan chain.
scan_clock_a	Defines the scan clock of the master latch for the LSSD scan style.
scan_clock_b	Defines the scan clock of the slave latch for the LSSD scan style.
shift_enable	Defines a <code>test_signal</code> object of type <code>shift_enable</code> .
shift_register_segment	Defines a shift register scan-chain segment object.
tap_port	Defines a TAP port (JTAG signal).
test_bus_port	Defines a test bus port.
test_clock	Defines a test clock object.
test_mode	Defines a <code>test_signal</code> object of type <code>test_mode</code>

### Related Information

Related commands:	<a href="#">define_dft_abstract_segment</a> on page 689 <a href="#">define_dft_boundary_scan_segment</a> on page 695 <a href="#">define_dft_dft_configuration_mode</a> on page 699 <a href="#">define_dft_domain_macro_parameters</a> on page 702 <a href="#">define_dft_fixed_segment</a> on page 704 <a href="#">define_dft_floating_segment</a> on page 706 <a href="#">define_dft_jtag_instruction</a> on page 708 <a href="#">define_dft_jtag_instruction_register</a> on page 712 <a href="#">define_dft_jtag_macro</a> on page 714
-------------------	---

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

[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 761  
[define dft test mode](#) on page 765

## **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

*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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

	This option must be specified with the <code>-skew_safe</code> option.
<code>-skew_latch</code>	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 <code>-skew_safe</code> option.
<code>-skew_safe</code>	Indicates whether the abstract segment has a data lockup element connected at the end of its scan chain.
<code>-tail_clock_off_state {high low}</code>	Specifies the off state of the system clock specified through the <code>-tail_clock_port</code> option.  <b>Note:</b> This option applies only to the clocked LSSD scan style.
<code>-tail_clock_port port</code>	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 <code>-clock_port</code> option.
<code>-tail_edge_rise   -tail_edge_fall</code>	Specifies the active edge of the clock specified through the <code>-tail_clock_port</code> option.  <b>Default:</b> <code>-tail_edge_rise</code>
<code>-tail_inversion</code>	Indicates an inversion at the scan data output pin of the abstract segment.
<code>-test_mode_active {low   high}</code>	Specifies the active value for the test-mode port.  This option should immediately follow the corresponding <code>-test_mode_port</code> option.
<code>-test_mode_port subport</code>	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

[define\\_dft scan\\_chain](#) on page 739

Affects these commands:

[check\\_dft\\_rules](#) on page 648

[connect\\_scan\\_chains](#) on page 681

[fix\\_scan\\_path\\_inversions](#) on page 775

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

[report dft chains on page 881](#)

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\_ACPSCLK test signal.

`-acpsen_11496 pin`

Specifies the name of the pin on the boundary scan segment that will be driven by the JTAG\_ACPSEN test signal.

`-acptrenbl_11496 pin`

Specifies the name of the pin on the boundary scan segment that will be driven by the JTAG\_ACTRENBL test signal.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<code>-acpulse_11496 pin</code>	Specifies the name of the pin on the boundary scan segment that will be driven by the JTAG_ACPULSE test signal.
<code>-bsdl_file file</code>	Specifies the file containing the BSDL abstract string for the boundary scan segment.
<code>-bsdl_string string</code>	Specifies the BSDL abstract string for the boundary scan segment.
<code>-capturedr capturedr_pin</code>	Specifies the name of the CAPTURE_DR pin on the boundary scan segment.
<code>-clockdr clockdr_pin</code>	Specifies the name of the CLOCK_DR pin on the boundary scan segment.
<code>-differential_pair {positive_leg_pin negative_leg_pin}</code>	Specifies the differential pair in the form of a positive leg pin name and negative leg pin name on the boundary scan segment.
<code>-highz highz_pin</code>	Specifies the name of the HIGHZ pin on the boundary scan segment
<code>-index bsr_position</code>	Specifies the relative position of the boundary scan segment in the BSR. Specify an integer value of zero or greater.
<code>-instance instance</code>	Specifies the instance name of the module for which the boundary scan segment is defined.
<code>-libcell cell</code>	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.
<code>-mode_a mode_a_pin</code>	Specifies the name of the MODE_A pin on the boundary scan segment

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-mode\_b mode\_b\_pin**

Specifies the name of the MODE\_B pin on the boundary scan segment

**-mode\_c mode\_c\_pin**

Specifies the name of the MODEC pin on the boundary scan segment

**-module subdesign**

Specifies the subdesign (module) to which the element belongs.

**-name segment\_name** Defines a name for the boundary scan segment.

**-shiftdr shiftdr\_pin**

Specifies the name of the SHIFT\_DR pin on the boundary scan segment.

**-tdi tdi\_pin**

Specifies the name of the TDI pin on the boundary scan segment.

**-tdo tdo\_pin**

Specifies the name of the TDO pin on the boundary scan segment.

**-updatedr updatedr\_pin**

Specifies the name of the UPDATE\_DR pin on the boundary scan segment.

### Example

- The following example defines an boundary scan segment using a set of differential port pairs.

```
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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- Each mode\_c pin on the boundary-scan segment will be connected to the JTAG\_MACRO JTAG\_INSTRUCTION\_DECODE\_MODE\_C output pin.

```
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 787

[write\\_bsdl](#) on page 903

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.

`-mode_enable_low test_signal...`

Name of the test signal(s) set to a logic\_0 value that controls the scan chains in the current mode.

`-name` Name of the scan mode.

`-type {scan | wrapper}`

Specifies the type of the scan mode.

`-usage {extest | intest | mission}`

Specifies the usage of the scan mode for IEEE 1500 core wrapper insertion.

This option is only valid with `-type wrapper`.

## Example

The following example defines scan mode `scanModeA`. Test signal `test1` is specified to have an active high logic value and test signal `test2` is specified to have an active low logic value in this mode of operation:

```
define_dft dft_configuration_mode -name scanModeA design \
    -mode_enable_high test1 -mode_enable_low test2
```

## Related Information

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 449

[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 900

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

[write\\_dft\\_abstract\\_model](#) on page 914

[write\\_et\\_atpg](#) on page 918

[write\\_et\\_bsv](#) on page 926

[write\\_et\\_mbist](#) on page 938

[write\\_et\\_rrfa](#) on page 944

[write\\_scandef](#) on page 964

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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: [iinsert\\_dft\\_opcg](#) on page 823

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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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_core7i_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 881

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
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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 881

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**

<code>-capture <i>string</i></code>	Specifies the values that must be captured into a register during the CaptureDR state.
<code>-design <i>design</i></code>	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.
<code>-length <i>integer</i></code>	Specifies the length of the custom test data register (TDR).
<code>-name <i>string</i></code>	Specifies the name of the user-defined instruction.
<code>-opcode <i>string</i></code>	Specifies the binary code for this instruction.
<code>-private</code>	Specifies that the defined instruction is private.
<code>-register <i>string</i></code>	Specifies the name of the custom test data register (TDR).

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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\_DECODE\_ *instruction* pin on the JTAG\_MACRO subdesign, where *instruction* is the name that you specified through the -name option.

**-register\_reset {pin|port}**

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_RESET pin on the JTAG\_MACRO subdesign.

**-register\_reset\_inverted**

Specifies that the JTAG\_RESET pin of the custom test register (TDR) has an active low polarity.

**-register\_runidle {pin|port}**

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_RUNIDLE pin on the JTAG\_MACRO subdesign.

**-register\_shiftdr {pin|port}**

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-register_tck {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_TCK pin on the JTAG\_MACRO subdesign.

`-register_tdi {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_TDI pin on the JTAG\_MACRO subdesign.

`-register_tdo {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_register\_TDO pin on the JTAG\_MACRO subdesign, where *register* is the name that you specified through the `-register` option

`-register_updatedr {pin|port}`

Specifies the name of the pin on the custom test data register (TDR) that must be connected to the JTAG\_UPDATEDR pin on the JTAG\_MACRO subdesign.

`-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 787

[insert\\_dft\\_jtag\\_macro](#) on page 806

[insert\\_dft\\_mbist](#) on page 817

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**

<code>-capture <i>string</i></code>	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.  <i>Default:</i> 01
<code>-design <i>design</i></code>	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.
<code>-length <i>integer</i></code>	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 <i>n</i> is the number of bits in the instruction register, a total of $2^n$ instructions can be defined including the four mandatory instructions.  <i>Default:</i> 2
<code>-name <i>string</i></code>	Specifies the name of the user-defined instruction register.

### **Examples**

- The following example defines instruction register `INSTR_REGISTER` with length 3. A register of length 3 allows you to create  $2^3$  instructions, or four user-defined instructions besides the four mandatory instructions.

```
define_dft jtag_instruction_register -name INSTR_REGISTER -length 3
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 787

[insert\\_dft\\_jtag\\_macro](#) on page 806

[insert\\_dft\\_mbist](#) on page 817

Related attributes: [JTAG Instruction Register Attributes](#)

## **define\_dft jtag\_macro**

```
define_dft jtag_macro
    [-module subdesign] [-libcell libcell]
    [-instance instance]
    [-bsr_shiftdr {pin|port|subport}]
    [-bsr_clockdr {pin|port|subport}]
    [-bsr_updatedr {pin|port|subport}]
    [-reset {pin|port|subport}]
    [-runidle {pin|port|subport}]
    [-shiftdr {pin|port|subport}]
    [-clockdr {pin|port|subport}]
    [-updatedr {pin|port|subport}]
    [-capturedr {pin|port|subport}]
    [-exit1dr_state {pin|port|subport}]
    [-mode_a {pin|port|subport}]
    [-mode_b {pin|port|subport}]
    [-mode_c {pin|port|subport}]
    -tdi {pin|port|subport} -tdo {pin|port|subport}
    -tck {pin|port|subport} -tms {pin|port|subport}
    [-trst pin/port|subport]
    [-boundary_tdo {pin|port|subport}]
    [-tdo_enable {pin|port|subport}]
    [-highz {pin|port|subport}]
    [-dot6_acdcsel {pin|port|subport}]
    [-dot6_acpulse {pin|port|subport}]
    [-dot6_preset_clock {pin|port|subport}]
    [-dot6_trcell_enable {pin|port|subport}]
    [-por {pin|port|subport}] [-name string]
```

Identifies a pre-instantiated JTAG Macro.

**Note:** To use this command you need an 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

**-tdo {pin|port|subport}**

Specifies the TAP controller TDO output pin on the JTAG macro.

**-tdo\_enable {pin|port|subport}**

Specifies the enable output pin which drives the JTAG TDO output enable pin.

**-tms {pin|port|subport}**

Specifies the TAP controller TMS input pin on the JTAG macro.

**-trst {pin|port|subport}**

Specifies the TAP controller TRST input pin on the JTAG macro.

**-updatedr {pin|port|subport}**

Specifies the update data register (UPDATEDR) output pin custom test data register.

## Examples

- The following example defines a third party TAP controller with a specified instance location.

```
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.

```
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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 806

## **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] 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**

<code>-design <i>design</i></code>	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.
<code>-hookup_period <i>integer</i></code>	Specifies the period interval at the hookup pin. The hookup period is specified in picoseconds.  <i>Default:</i> value of <code>-period</code> option
<code>-hookup_pin <i>pin</i></code>	Specifies the core-side hookup pin to be used by the <code>insert_dft mbist</code> command to make the MBIST (PMBIST) clock connection.  <b>Note:</b> 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

`-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.

```
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 817

[insert\\_dft pmbist](#) on page 825

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 <i>string</i> | Defines the functionality of the direct access pin. The option can have one of the following values: <ul style="list-style-type: none"><li>■ <code>burnin_run</code><br/>This signal must be held inactive for a minimum of three MBIST clock periods at the start of the test sequence. Once activated, the <code>burnin_run</code> 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 <code>monitor</code> during the course of the execution.</li><li>■ <code>device_schedule_serial</code><br/>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.<br/>The execution occurs in parallel when the signal is inactive or undefined.</li></ul> |

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `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](#) on page 817

Sets these attributes:      [Direct Access Function Attributes](#)

## **define\_dft opcg\_domain**

```
define_dft opcg_domain [-name name]
    -osc_source osc_source
    -domain_macro_parameter domain_macro_parameter
    -opcg_trigger opcg_trigger
    -location {subport | port | pin}
    -min_domain_period float
    [-divide_by integer]
    [-scan_clock scan_clock] [-design design]
```

Defines an OPCG (clock) domain.

The command returns the directory path to the `opcg_domain` object that it creates. You can find the objects created by the `define_dft opcg_domain` constraints in:

```
/designs/top/dft/opcg/opcg_domains/
```

### **Options and Arguments**

<code>-design design</code>	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.
<code>-divide_by integer</code>	Specifies to build an internal clock divider circuit which divides the oscillator source frequency by the specified number.
<code>-domain_macro_parameter domain_macro_parameter</code>	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 <code>define_dft domain_macro_parameters</code> command.
<code>-location {subport   port   pin}</code>	Specifies where to insert the domain macro. Specify an existing top-level port, subport, or hierarchical pin name.
<code>-min_domain_period float</code>	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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- The following command inserts the domain macro after pin `out1` of subdesign instance `i_core`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/out1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro within the subdesign for instance `i_core` prior to output pin `out1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/subports_out/out1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro before the `in1` pin of subdesign instance `i_core`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/in1 -min_domain_period 1000 -divide_by 6
```

- The following command inserts the domain macro within the subdesign for instance `i_core` after the input pin `in1`.

```
define_dft opcg_domain -name OD1 -osc_source OSC1 \
-domain_macro_parameter DOMAIN_MACRO_PARAMETER_1 -opcg_trigger TRIGGER1 \
-location i_core/subports_in/in1 -min_domain_period 1000 -divide_by 6
```

### Related Information

#### [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 823

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**

<code>-design <i>design</i></code>	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.
<code>-jtag_controlled</code>	Specifies whether a JTAG instruction is used to lock the PLLs for OPCG operation.
<code>-mode_init <i>file</i></code>	Specifies the name of file containing the OPCG mode initialization sequence that ensures that PLLs are properly initialized and locked on the input oscillators.
<code>-name <i>name</i></code>	Specifies the <code>opcg_mode</code> object name of the OPCG mode.  <i>Default:</i> <code>DFT_prefix_opcg_mode_n</code>
<code>-osc_source_parameters <i>string</i></code>	Specifies the oscillator source parameters specific for this OPCG mode.  Use the following format:  <code>{<i>name</i> <i>osc_source_output_period</i> <i>ref_clock_period</i>}</code>  where <i>name</i> is the name of the <code>osc_source</code> 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 opcgMode1.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 918

Related constraint: [define\\_dft osc\\_source](#) on page 732

Sets these attributes: [OPCG Mode Attributes](#)

[Osc Source Reference Attributes](#)

## **define\_dft opcg\_trigger**

```
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 unqualified 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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-osc_source osc_source`

Specifies the object name of the oscillator source whose trigger macro is enabled by the defined signal.

The oscillator source must have been defined with `define_dft osc_source` command.

`{pin|port}`

Specifies the driving (input) pin or port for the OPCG trigger signal.

**Note:** If multiple designs are loaded and you did no specify the `-design` option, you can also specify the full path to the driver.

`-scan_clock scan_clock`

Specifies the test clock that must be used for shifting the OPCG trigger logic in fullscan mode.

If you omit this option, the scan clock specified with the `insert_dft opcg` command is used instead.

### 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 823

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.

<code>-min_output_period float</code>	Specifies the minimum period of the generated (output) clock. Specify the clock period in picoseconds.
<code>-name osc_source</code>	Specifies the name of the <code>osc_source</code> object for the oscillator source.
	<i>Default:</i> OSC_SOURCE_n
<code>pin</code>	Specifies the output pin of the PLL.
<code>-ref_clock_pin pin</code>	Specifies the reference (input) clock pin or port for the PLL.

## Example

The following command defines an oscillator source for instance `instPLL1` whose reference (input) clock has a period between 20000 and 4000 ps (or clock frequency between 50 and 250 MHz) and whose output clock has a period between 2000 and 500 ps (or a clock frequency between 500 and 2000 MHz).

```
define_dft osc_source -name instPLL1
  -ref_clock_pin REFCLKPORT1 \
  -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 823

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 825](#)

Sets these attributes: [Programmable Direct Access Function Attributes](#)

## **define\_dft preserved\_segment**

```
define_dft preserved_segment [-name segment_name]
  { {instance|segment_name}... [-sdi pin] [-sdo pin]
  | -analyze -sdi {pin|port} -sdo {pin|port} }
  [ -connected_shift_enable
  | [-connected_scan_clock_a] [-connected_scan_clock_b] ]
  [-allow_reordering]
```

Defines a preserved segment. In a preserved segment, the elements of the mapped segment are already connected in the specified order, and they cannot be reordered by a physical scan reordering tool.

A preserved segment is a user-specified scan segment which can be associated with either

- A user-defined top-level chain—created using the [define\\_dft scan\\_chain](#) command
- A tool-created scan chain—created using the [connect scan chains](#) command

The command returns the directory path to the object that it creates. You can find the objects created by the `define_dft preserved_segment` constraints in:

`/designs/top_design/dft/scan_segments`

### **Options and Arguments**

`-allow_reordering`      Indicates whether the order of the elements can be changed during scan configuration and by a physical scan reordering tool.

`-analyze`      Analyzes the connectivity of the scan segment, and returns the elements of the segment given its endpoints.

**Note:** If the segment includes degenerated scan flops in its element list, you must also specify the `-connected_shift_enable` option for the command to complete successfully.

`-connected_scan_clock_a` (`-connected_scan_clock_b`)

Indicates that the `scan_clock_a` (`scan_clock_b`) port of the module boundary is driven by external logic (preconnected). The external logic connected to the `scan_clock_a` (`scan_clock_b`) pin of the module will not be modified by the scan configuration engine.

This option applies only for the clocked LSSD scan style.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 (hierarchical) 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/SDa
```

- 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 881

Sets these attributes: [Scan Segment Attributes](#)

## **define\_dft scan\_chain**

```
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**

<code>-analyze</code>	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.
<code>-body segment</code>	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 <code>define_dft xxx_segment</code> constraint, where <code>xxx</code> is either <code>abstract</code> , <code>fixed</code> , <code>floating</code> , <code>preserved</code> , or <code>shift_register</code> .
<code>-complete</code>	Specifies that the defined chain is complete and no other flip-flops should be added to the chain.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-dont_overlay	<p>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 <code>write_scandef</code> 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.</p>
-edge {rise fall}	<p>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 <code>-domain</code>.</p> <p>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.</p> <p>This option is ignored if you enabled the <code>dft_mix_clock_edges_in_scan_chains</code> attribute.</p> <p><b>Note:</b> This option only applies to the muxed scan style.</p>
-head <i>segment</i>	<p>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 <code>define_dft xxx_segment</code> constraint, where <code>xxx</code> is either <code>abstract</code>, <code>fixed</code>, <code>floating</code>, <code>preserved</code>, or <code>shift_register</code>.</p>
-hookup_pin_sdi <i>pin</i>	<p>Specifies the core-side hookup pin to be used for the scan data input signal during scan chain connection.</p> <p><b>Note:</b> 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.</p>
-hookup_pin_sdo <i>pin</i>	<p>Specifies the core-side hookup pin to be used for the scan data output signal during scan chain connection.</p>

**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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**Default:** The default shift-enable signal or chain-specific shift-enable signal is used as the select control signal to the mux.

{-shared\_output | -non\_shared\_output}

Specifies whether an existing functional output port can be used as scan data output port. If the functional port can be used for scan data purposes, a mux is inserted in the scan data path by the `connect_scan_chains` command.

One of these options must be specified when the specified scan data output signal is already connected in the circuit.

-shift\_enable *test\_signal*

Designates a chain-specific shift-enable port or pin.

If you omit this option, the default shift-enable signal specified using a `define_dft shift_enable` constraint is used.

-tail *segment*

Indicates that the specified segment (an ordered set of scan flip-flops) must be placed at the tail (closest to the scan data output) of the scan chain. The segment must have been previously defined with a `define_dft xxx_segment` constraint, where *xxx* is either `abstract`, `fixed`, `floating`, `preserved`, or `shift_register`.

-terminal\_lockup {level\_sensitive | edge\_sensitive}

Specifies the type of lockup element that configuration can insert at the tail end of the chain to connect to the specified scan data output signal.

If this option is not specified, no terminal lockup element will be inserted.

**Note:** This option only applies to the muxed scan style.

## Examples

- The following example creates a chain containing 3 segments previously defined:

```
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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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 761

Affects these commands:

[connect\\_scan\\_chains](#) on page 681  
[report\\_dft\\_chains](#) on page 881

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] [-no_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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

`-name name`

Specifies the `test_signal` object name of the `scan_clock_a` 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-no_ideal	Marks the <code>scan_clock_a</code> signal as non-ideal. This allows buffering of the <code>scan_clock_a</code> network during optimization.  By default, the <code>scan_clock_a</code> signal is marked ideal.  <b>Note:</b> If the test signal is marked as ideal, RTL Compiler sets the <code>ideal_network</code> attribute to <code>true</code> on the pin or port for the <code>scan_clock_a</code> signal
-period <i>integer</i>	Together with the <code>-divide_period</code> option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing <code>-period</code> by <code>-divide_period</code> .  <i>Default:</i> 50000 (20 MHz test clock)
-rise <i>integer</i>	Together with the <code>-divide_rise</code> 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 <code>-rise</code> by <code>-divide_rise</code> .  <i>Default:</i> 50

### Example

- The following example defines `sca` as the driver for the `scan_clock_a` signal and assigns `SCA` as the `test_signal` object name.

```
define_dft scan_clock_a -name SCA sca
```

### Related Information

[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 881

[write\\_atpg](#) on page 900

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] [-no_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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 controlability of any logic between the top-level `scan_clock_b` signal and its designated hookup pin under test-mode setup.

`-hookup_polarity {inverted|non_inverted}`

Specifies the polarity of the `scan_clock_b` signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-no_ideal	Marks the <code>scan_clock_b</code> signal as non-ideal. This allows buffering of the <code>scan_clock_b</code> network during optimization.  By default, the <code>scan_clock_b</code> signal is marked ideal.  <b>Note:</b> If the test signal is marked as ideal, RTL Compiler sets the <code>ideal_network</code> attribute to <code>true</code> on the pin or port for the <code>scan_clock_b</code> signal
-period <i>integer</i>	Together with the <code>-divide_period</code> option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing <code>-period</code> by <code>-divide_period</code> .  <i>Default:</i> 50000 (20 MHz test clock)
-rise <i>integer</i>	Together with the <code>-divide_rise</code> 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 <code>-rise</code> by <code>-divide_rise</code> .  <i>Default:</i> 70

### Example

- The following example defines `sca` as the driver for the `scan_clock_b` signal and assigns `SCB` as the `test_signal` object name.

```
define_dft scan_clock_b -name SCB scb
```

### Related Information

[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 881

[write\\_atpg](#) on page 900

Sets these attributes: [Test Signal Attributes](#)

Affected by this attribute: [dft\\_scan\\_style](#)

## **define\_dft shift\_enable**

```
define_dft shift_enable [-name name] -active {low|high}
    [-default] [-no_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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<code>-design <i>design</i></code>	<p>Specifies the name of the design for which the shift enable is defined.</p> <p>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.</p>
<code>-hookup_pin <i>pin</i></code>	<p>Specifies the core-side hookup pin to be used for the top-level shift-enable signal during DFT synthesis.</p> <p><b>Note:</b> When you specify this option, the RC-DFT engine does not validate the controlability of any logic between the top-level shift_enable signal and its designated hookup pin under test-mode setup.</p>
<code>-hookup_polarity {non_inverted   inverted}</code>	<p>Specifies the polarity of the shift-enable signal at the core-side hookup pin.</p> <p>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.</p>
<code>-name <i>name</i></code>	<p>Specifies the <code>test_signal</code> object name of the shift-enable signal.</p> <p>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.</p>
<code>-lec_value {auto   0   1   no_value }</code>	<p>Specifies the approach to constraining the pin for LEC validation by the <code>write_do_lec</code> command. You can specify any of the following values:</p> <ul style="list-style-type: none"><li>■ <code>auto</code>—Writes the opposite of the test mode active value as the “add pin constraint” in the do file</li><li>■ <code>0</code>—Writes a logic 0 for the <code>add pin</code> constraint in the do file</li><li>■ <code>1</code>—Writes a logic 1 for the <code>add pin</code> constraint in the do file</li><li>■ <code>no_value</code>—Does not write the signal as an “add pin constraint” in the do file</li></ul>

*Default:* auto

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-no_ideal	Marks the shift-enable signal as non-ideal. This allows buffering of the shift-enable network during optimization.  <i>Default:</i> The shift-enable signal is marked ideal.  <b>Note:</b> If the test signal is marked as ideal, RTL Compiler sets the <code>ideal_network</code> attribute to <code>true</code> on the pin or port for the shift-enable signal
{pin port}	Specifies the driving pin or port for the shift-enable signal.  <b>Note:</b> If multiple designs are loaded and you did no specify the <code>-design</code> option, you can also specify the full path to the driver.

#### Example

- When the following constraint is given, the `check_dft_rules` command propagates a logic1 from the `p_top/SE` pin into the design.

```
define_dft shift_enable -active low -hookup_pin p_top/SE -hookup_polarity inverted
```

#### Related Information

[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 843

[report\\_dft\\_chains](#) on page 881

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 <i>instance</i>	Specifies the last flop in the shift register. Specify the hierarchical instance name of the flop.
-name <i>segment_name</i>	Defines a name for the segment that you can use to reference in the <a href="#"><u>define_dft scan_chain</u></a> constraint.
-start_flop <i>instance</i>	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 881

Related command: [identify\\_shift\\_register\\_scan\\_segments](#) on page 779

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**

<code>-create_port</code>	Creates the port if it does not exist.
<code>design</code>	Specifies the design for which the tap ports are being defined.
<code>-hookup_pin {pin   port}</code>	Specifies the core-side hookup pin to be used for the top-level JTAG signal during DFT synthesis.
<code>-hookup_polarity {non_inverted   inverted}</code>	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.
<code>-no_trst_test_signal</code>	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.
<code>{pin   port}</code>	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**

```
define_dft test_bus_port
  [-name string] -function string [-index integer]
  [-hookup_pin {pin|port}] [-hookup_polarity string]
  [-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**

<code>-create_port</code>	Creates the port if it does not exist.
<code>-function <i>string</i></code>	Specifies the function of the test bus port. Following values can be specified
<b>Function</b>	<b>Purpose</b>
<code>compress_sdi</code>	Common scan data input (broadcast to multiple cores)
<code>compress_sdo</code>	Common scan data output
<code>serial_sdi</code>	1500 Wrapper Serial Input
<code>serial_sdo</code>	1500 Wrapper Serial Output
<code>select_wir</code>	Select WIR access control signal
<code>select_serial</code>	Select 1500 serial mode
<code>select_bypass</code>	Select 1500 bypass mode
<code>compression_clock</code>	Test clock used for compression mask (and MISR if present)
<code>compression_enable</code>	Control to enable compression mode
<code>spread_enable</code>	Control to enable the xor decompressor

## Command Reference for Encounter RTL Compiler

### Design for Test

mask_load	Control to enable mask loading mode
mask_enable	Data input to enable channel masking
mistr_reset_enable	Control signal to reset the MISR (mistr compression only)
wint	Wrapper Intest control signal
wext	Wrapper Extest control signal
wrapper_reset	Wrapper instruction register reset control
wrapper_clock	Wrapper instruction and dedicated wrapper cell clock
wrapper_and_compression_clock	Shared compression and wrapper clock
shift_wr	WIR shift control
capture_wr	WIR capture control
update_wr	WIR update control
custom	User specified control signal

**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.

`-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.

## Related Information

[Hierarchical Test Flow: Preparing a Core in Design for Test in Encounter RTL Compiler](#)

[Inserting Core-Wrapper Logic in Design for Test in Encounter RTL Compiler](#)

Affects this command:

[insert\\_test\\_compression](#) on page 867

Sets these attributes:

[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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**Note:** When you specify this option, the RC-DFT engine does not validate the controllability of any logic between the top-level hookup pin under test-mode setup.

`-hookup_polarity {inverted | non_inverted}`

Specifies the polarity of the test clock signal at the core-side hookup pin.

By default, a non inverted polarity is assumed, but if the assumed value at the hookup pin conflicts with the propagated value, the tool will issue an error message.

`-name test_clock`

Specifies the name of the test clock that is being defined.  
Each clock object in your design must have a unique name. If you define a new test clock with the same name as an existing clock, an error message will be issued.

**Note:** The clock name allows you to search for the clock later (through the `find` command) or to recognize it in reports.

`-period integer`

Together with the `-divide_period` option, determines the clock period interval. The clock period is specified in picoseconds and is derived by dividing `-period` by `-divide_period`.

*Default:* 50000 (20 MHz test clock)

`{pin|port}`

Specifies the test clock input pin or port.

If you specify multiple pins, these pins are assumed to have zero skew: they can be mixed without lockup latches.

`-rise integer`

Together with the `-divide_rise` option, determines the time that the rising edge occurs with respect to the beginning of the clock period. The time is specified as a percentage of the period and is derived by dividing `-rise` by `-divide_rise`.

*Default:* 50

## Example

The following example defines three test clocks, four test clock ports and two DFT clock domains.

```
define_dft test_clock -name CLK1X -domain domain_1 -period 20000 CLK1
define_dft test_clock -name CLK2X -domain domain_1 -period 20000 CLK2 CLK2b
define_dft test_clock -name CLK3X -domain domain_2 -period 20000 CLK3
```

The four test clock ports are CLK1, CLK2, CLK2b, and CLK3. Test clock CLK2X comprises equivalent test clocks CLK2 and CLK2B (they can be mixed in the same scan chain without any lockup element). Test clocks CLK1X, CLK2X belong to the same DFT clock domain and are compatible (requires lockup elements between compatible chain segments triggered by the different test clocks). Test clock CLK3X belongs to its own domain.

## Related Information

See the following sections in *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 771

[report\\_dft\\_chains](#) on page 881

Sets these attributes:      [Test Clock Attributes](#)

## **define\_dft test\_mode**

```
define_dft test_mode [-name name] -active {low | high}
    [-no_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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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.

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-no_ideal	Marks the test-mode signal as non-ideal. This allows buffering of the test-mode network during optimization.  <i>Default:</i> The test-mode signal is marked ideal.  <b>Note:</b> If the test signal is marked as ideal, RTL Compiler sets the <code>ideal_network</code> attribute to <code>true</code> on the pin or port for the test signal.
{pin port}	Specifies the driving pin or port for the test signal.  <b>Note:</b> If multiple designs are loaded and you did no specify the <code>-design</code> 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.
 <i>Important</i>	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 <code>write_dolec</code> dofile. Not specifying this option for the appropriate test signals will result in over constraining the <code>write_dolec</code> 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.  <b>Note:</b> This option cannot be specified with the <code>-create_port</code> 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 771

[insert\\_dft\\_shadow\\_logic](#) on page 843

[insert\\_dft\\_test\\_point](#) on page 849

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**

<code>-mode <i>integer</i></code>	Specifies the mode for tracing back. <ul style="list-style-type: none"><li>■ 0 does not perform constant propagation</li><li>■ 1 performs tied-constant propagation</li><li>■ 2 performs tied-constant and test-mode propagation</li><li>■ 3 performs tied-constant, test-mode and shift-enable propagation</li></ul> <p><i>Default:</i> 3</p>
<code>-continue</code>	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.
<code>{pin port}</code>	Specifies the pin or port from which to start the trace back.
<code>-polarity</code>	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.
<code>-print</code>	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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-dont_check_dft_rules`

Prevents the DFT rules from being checked automatically after fixing violations.

`-dont_map`

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

`-exclude_xsource instance`

Specifies instances to exclude from automatic fixing of X-source violations.

`-insert_observe_scan`

Inserts a flip-flop for observability. If you fix DFT violations in a generic netlist, the flip-flop is mapped to a scan flip-flop during synthesis. If you fix DFT violations in a mapped netlist, the flip-flop is mapped to a scan flip-flop. You need to rerun the `check_dft_rules` command to update the DFT status of the observability flop prior to connecting it in a scan chain.

**Note:** This option can only be used when fixing an asynchronous set reset violation and requires the `-test_clock_pin` option.

`{pin | port}`

Specifies the test signal pin or port to use to control the set or reset. By default, the set or reset are controlled by the `-test_control` option.

To specify this option, the pin or port must first be declared as a `test_mode` signal using the `define_dft test_mode` constraint.

`-preview`

Reports how the violations will be fixed, without making modifications to the netlist.

`[-rise | -fall]`

Specifies to use the rising or falling edge of the test clock to fix the DFT violation during test-mode operation.

*Default:* `-rise`

`-test_clock_pin {pin | port}`

Specifies the test clock signal to be used. Specify the pin or port from where the clock signal originates. In most applications, the clock pin or port is identified when checking the DFT rules.

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.

```
fix_dftViolations -clock -async_set -async_reset  
-test_control tm -insert_observe_scan -test_clock_pin CK1 -rise
```

If you do not want to use the same test clock to fix the clock violations and to drive the clock pin of the observation flip-flops, you need to enter two commands. For example,

```
fix_dftViolations -clock -test_control tm  
fix_dftViolations -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 asynch reset violations using test mode signal `tm` and test clock `CK1` using the rising edge as the active edge.

```
fix_dft_violations -clock -async_set -async_reset  
-test_control tm -test_clock_pin CK1 -rise
```

- The following example instructs 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 761

[define\\_dft test\\_mode](#) on page 765

Affects these commands: [check\\_dft\\_rules](#) on page 648

[report\\_dft\\_registers](#) on page 884

[report\\_dftViolations](#) on page 886

[synthesize](#) on page 379

Sets these attributes: [dft\\_status](#)

[dftViolation](#)

[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**

<i>design</i>	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 823

## **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**

<code>-design design</code>	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.
<code>-dont_check_dft_rules</code>	Prevents the DFT rules from being automatically checked after identifying abstract segments for multi-bit scan cells.
<code>-libcell libcell</code>	Specifies the multi-bit library cells on which to perform abstract segment identification.
<code>-preview</code>	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:

```
rc:/> identify_multibit_cell_abstract_scan_segments -preview
```

Would execute command:

```
define_dft abstract_segment -length 1 -sdi SI1 -sdo Q1  
-shift_enable_port SE1 -active high -clock_port CP -rise  
-libcell/libraries/tcbn65ulp_c070701wc2/libcells/DUALSDFQD0 -name DUALSDFQD0
```

Would execute command:

```
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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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**

```
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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**Note:** This option cannot be specified with the `-stil`, `-mode_init`, `-macro`, `-library`, `-et_log` and `-continue_with_severe_warnings` options.

<code>-library string</code>	Specifies the list of Verilog structural library files. Specify the list in a quoted string. Refer to the <code>write_et_atpg -library</code> option description for additional information.
	<b>Note:</b> This option is only required when you invoke this command on a mapped netlist.
<code>-macro string</code>	Specifies the name of the macro in the STIL file that contains the initialization vectors to be simulated.
	<i>Default:</i> <code>test_setup</code>
<code>-mode_init file</code>	Specifies the name of the file that contains the mode initialization sequence in TBDpatt format.
<code>-preview</code>	Reports the fixed value registers but does not set the test mode values.
<code>-stil file</code>	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
-----

Test Function      Block Name
+TI               tc/ts_reg[0]
+TI               tc/ts_reg[1]
+TI               tc/ts_reg[2]
-----
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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

- The following command creates the test signals and automatically reruns the DFT rule checker.

```
rc:/> 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

...
rc:/> 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 765

[define\\_dft\\_test\\_clock](#) on page 761

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 | rrfra_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 <code>-config_file</code> .
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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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 <code>wrapper</code> .

### Related Information

Related commands:	<a href="#">insert_dft_boundary_scan</a> on page 787 <a href="#">insert_dft_dfa_test_points</a> on page 802 <a href="#">insert_dft_jtag_macro</a> on page 806 <a href="#">insert_dft_lockup_element</a> on page 810 <a href="#">insert_dft_logic_bist</a> on page 811 <a href="#">insert_dft_mbist</a> on page 817 <a href="#">insert_dft_opcg</a> on page 823 <a href="#">insert_dft_pmbist</a> on page 825 <a href="#">insert_dft_ptam</a> on page 830 <a href="#">insert_dft_rrfa_test_points</a> on page 833 <a href="#">insert_dft_scan_power_gating</a> on page 840 <a href="#">insert_dft_shadow_logic</a> on page 843 <a href="#">insert_dft_shift_register_test_points</a> on page 848
-------------------	---

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

[insert\\_dft test point](#) on page 849  
[insert\\_dft user test point](#) on page 855  
[insert\\_dft wrapper cell](#) on page 857  
[insert\\_dft wrapper instruction register](#) on page 863  
[insert\\_dft wrapper mode decode block](#) on page 865

## **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]
    [-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

*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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<code>-tdo port</code>	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.  <b>Note:</b> An existing TDO port must have a tristate I/O pad.  <i>Default:</i> TDO
<code>-tms port</code>	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.  <i>Default:</i> TMS
<code>-trst port</code>	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.  <i>Default:</i> TRST

### 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 761

[define\\_dft\\_test\\_mode](#) on page 765

[compress\\_scan\\_chains](#) on page 660

[insert\\_dft\\_mbist](#) on page 817

[insert\\_dft\\_ptam](#) on page 830

Affects these commands: [compress\\_scan\\_chains](#) on page 660

[insert\\_dft\\_mbist](#) on page 817

[insert\\_dft\\_ptam](#) on page 830

Sets these attributes: [boundary\\_type](#)

[dft\\_jtag\\_macro\\_exists](#)

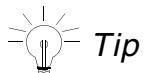
[JTAG Port Attributes](#)

## **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]
        [-mask_sharing_ratio integer]
        [-apply_timing_constraints
            [-timing_mode_names mode_list]]
        [-write_timing_constraints file]
        [-low_pin_compression
            [-lpc_control shift_enable]
            [-shift_enable shift_enable]]]
    | -compressor misr
        [-serial_misr_read]
        [-misr_observe test_signal]
        [-misr_clock {port|pin}]
        {-misr_reset_enable test_signal
         | -misr_reset_clock test_signal}
        [-misr_read test_signal
         | -use_all_scan_ios_unidirectionally]
        [-misr_shift_enable test_signal]
        [-mask_sharing_ratio integer]
        [-mask {wide0 | wide1 | wide2}]
        [-mask_clock {port|pin}] [-mask_load test_signal]
        [-mask_enable test_signal_list]
    | -compressor hybrid
        [-serial_misr_read]
        [-misr_observe test_signal]
        [-misr_bypass test_signal]
        [-misr_clock {port|pin}]
        {-misr_reset_enable test_signal
         | -misr_reset_clock test_signal}
        [-misr_shift_enable test_signal]
        [-mask {wide0 | wide1 | wide2}]
        [-mask_sharing_ratio integer]
        [-mask_clock {port|pin}] [-mask_load test_signal]
        [-mask_enable test_signal_list] }
    [-dont_exploit_bidi_scanio] [-inside instance] [-preview]
    [-jtag_control_instruction jtag_instruction]
        [-allow_multiple_jtag_control]]
```

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<i>design</i>	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.
<i>-inside instance</i>	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.
<i>-jtag_control_instruction jtag_instruction</i>	Specifies which JTAG instruction will be used to target the compression macro's test data register.
<i>-lpc_control shift_enable_signal</i>	Specifies a shift-enable signal used to control low pin count compression.  You cannot specify the same shift-enable signal for both the <i>-lpc_control</i> and <i>-shift_enable</i> 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 <i>-auto_create</i> option is specified.
<i>-low_pin_compression</i>	Reduces the number of compression control pins required by using encoded control signals.
<i>-mask {wide0   wide1   wide2}</i>	Inserts scan channel masking logic of the specified type.  The masking types that can be used depend on the compressor type specified with the <i>-compressor</i> option.  By default, no masking logic is inserted.  <b>Note:</b> The syntax indicates which types are available for each of the compressor types.
<i>-mask_clock {pin port}</i>	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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-use_user_scan_chains scan_chains`

Specifies which scan chain definitions (from the `scan_chains` directory) to use when building the full scan chains.

In this case, the tool will use the port and chain names defined with the `define_dft scan_chain` commands.

`-write_timing_constraints file`

Specifies the file to which to write the timing constraints applied to the appropriate compression control signals to prevent the mapper from considering these paths for timing optimization. The timing constraints are not written if you specify the `-preview` option.

Timing constraints will be applied in all user-specified timing modes.

**Note:** If your design has multiple timing modes but you did not specify the `-timing_mode_names` option to list the timing modes for which to write the constraints, constraints for all modes are written to the file.

## Examples

In the following examples, assume that the design has 55 existing scan chains, but only five top-level scan chains are desired.

- The following command inserts compression logic in the design and auto creates the compression channels.

```
insert_dft compression_logic -build_new_scan_chains 5 -auto_create
```

- In the following example, the user first defines the five top-level chains. Next these chains are used by the `insert_dft compression_logic` command.

```
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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 881

Sets these attributes:

[compressed](#)

[dft\\_compression\\_signal](#)

[dft\\_mask\\_clock](#)

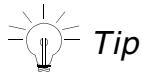
[dft\\_misr\\_clock](#)

[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.



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**

<code>-control_only</code>	Specifies to insert only control test points.
<code>design</code>	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.
<code>-fault_threshold integer</code>	Specifies to insert only those test point locations whose fault count is greater than or equal to the number specified.  <i>Default:</i> 0
<code>-gate_clock</code>	Enables the insertion of combinational logic to clock gate the test clocks of the inserted test points.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.tpf \
    -test_control test_mode1 \
    -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.tpf \
    -min_slack 1000 \
    -test_control test_model \
    -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.tpf \
    -test_control test_model \
    -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 761  
                                        [define\\_dft test\\_mode](#) on page 765

## **insert\_dft jtag\_macro**

```
insert_dft jtag_macro [-boundary_type {IEEE_11491|IEEE_11496}]\n    [-dont_map] [-jtag_macro_location instance]\n    [-insert_without_pad_logic [-create_ports]\n        [-dont_create_DFT_TDO_enable_port]]\n    [-preserve_tdo_connection]\n    [-tck port] [-tdi port] [-tdo port] [-tms port]\n    [-trst port] [-no_trst_test_signal]\n    [-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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.

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**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.

<code>-tck port</code>	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.  <i>Default:</i> TCK
<code>-tdi port</code>	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.  <i>Default:</i> TDI
<code>-tdo port</code>	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.  <b>Note:</b> An existing TDO port must have a tristate I/O pad.  <i>Default:</i> TDO
<code>-tms port</code>	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.  <i>Default:</i> TMS
<code>-trst port</code>	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.  <i>Default:</i> 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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 817

[insert\\_dft\\_ptam](#) on page 830

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_nedge] [-direct_bist_done_output_port]
  [-direct_bist_pass_output_port] [-direct_bist_fail_output_port]
  [-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_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]
  [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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<code>-add_masking</code>	Inserts the LBIST macro including the optional channel masking logic (only available with SETBIST support).
<code>-bist_clock port</code>	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.
<code>-bist_clock_hookup pin</code>	<p>Specifies the hookup pin for the LBIST oscillator clock. Usually the PLL output is used as hookup for the LBIST oscillator clock.</p>
<code>-capture_window integer</code>	Specifies the default value of the capture window.
<code>-capture_window_capture_value integer</code>	Specifies the value of the capture window (down) counter at which the (first) capture clock pulse is issued.
<code>-capture_window_counter_length integer</code>	Specifies the length of the capture window counter that is part of the LBIST macro.
<code>-capture_window_launch_value integer</code>	Specifies the value of the capture window (down) counter at which the launch clock pulse is issued.
<code>-control_inputs_0 port_list</code>	<p>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.</p>
<code>-control_inputs_1 port_list</code>	<p>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.</p>
<code>design</code>	Inserts the LBIST logic in the specified top-level design.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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\_nedge**  
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 do not need to be intercepted in LBIST mode.

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **Related Information**

##### [Inserting LBIST Logic in Design for Test in Encounter RTL Compiler](#)

Affected by these constraints: [define\\_dft\\_test\\_mode](#) on page 765

Related commands: [write\\_et\\_lbist](#) on page 934

[write\\_logic\\_bist\\_macro](#) on page 950

## **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]
  [-interface_file_dirs string] [-dont_map]
  [-dont_check_dft_rules] [-dont_check_mbist_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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

<code>-connect_to_jtag</code>	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.
<code>-continue_mbist string</code>	Specifies the CONTINUE_MBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.  <i>Default:</i> CONTINUE_MBIST
<code>-design design</code>	Specifies the name of the top-level design.
<code>-dft_configuration_mode mode</code>	Specifies the configuration mode in which MBIST will be operating. MBIST test pin verification and netlist back-tracing using the <code>check_mbist_rules</code> command must be done in this mode as well.  <i>Default:</i> mbist
<code>-diagnose_mbist string</code>	Specifies the DIAGNOSE_MBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.  <i>Default:</i> DIAGNOSE_MBIST
<code>-diagnose_rombist string</code>	Specifies the DIAGNOSE_ROMBIST instruction that must be loaded into the IEEE 1149.x TAP controller to access the MBIST engines.  <i>Default:</i> DIAGNOSE_ROMBIST
<code>-direct_access_only</code>	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.
<code>-directory directory_path</code>	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 <code>-interface_file_dirs</code> option, when this command is run on a higher level of the hierarchy.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

*Default: current\_working\_directory/  
mbist\_design*

**-dont\_check\_dft\_rules**

Prevents the DFT rules from being automatically checked after MBIST insertion.

**-dont\_check\_mbist\_rules**

Prevents the MBIST rules from being automatically checked after MBIST insertion.

**-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 `-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.

**-dont\_map**

Prevents the inserted logic from being mapped even if the design is already mapped to the target library.

**-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 `insert_dft mbist` command when processing the larger design. Separate the directory names with blank spaces.

**-measure\_ports ports**

Specifies the tester-accessible ports to be used as bitmap pattern measure ports in addition to the default JTAG TDO port.

**Note:** This option applies to MBIST bitmap patterns and chip-level designs only.

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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 `t_em`. 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 765

## Command Reference for Encounter RTL Compiler

### Design for Test

---

Affects these commands:	<a href="#">insert_dft_boundary_scan</a> on page 787 <a href="#">insert_dft_jtag_macro</a> on page 806 <a href="#">write_do_lec</a> on page 259 <a href="#">write_sdc</a> on page 294
Related commands:	<a href="#">check_mbist_rules</a> on page 654 <a href="#">write_dft_rtl_model</a> on page 917 <a href="#">write_et_bsv</a> on page 926 <a href="#">write_et_mbist</a> on page 938 <a href="#">write_mbist_testbench</a> on page 954
Related attributes:	<a href="#">dft_rtl_insertion</a> <a href="#">mbist_instruction_set</a>

## **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 test_signal`

Specifies the test signal to be used to gate the OPCG enable.

The test signal must have been previously defined by a  
`define_dft test_mode` constraint.

## Example

```
insert_dft opcg -opcg_enable OPCGMODE -opcg_load_clock opcg_load_clk \
-scan_clock scanclk -test_enable testmode
```

## Related Information

[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 761

[define\\_dft test\\_mode](#) on page 765

## **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]]
  [-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-direct\_access\_only** Specifies whether to insert PMBIST without the support from the JTAG Macro. The memory testing is controlled and examined directly from the PMBIST direct access ports/pins without using the JTAG macro. In this case, any information for the JTAG macro is ignored even when a JTAG macro is specified in the design.

**-directory directory\_path**

Specifies the directory in which PMBIST must create all intermediate files for its internal use.

*Default:* current\_working\_directory/  
pmbist\_design

**-dont\_create\_pmbist\_ports**

Prevents creation of PMBIST control ports for subsequent JTAG macro and direct access connection at the top-level instance.

Use this option when the instance is the top level of the chip, and you have not yet instantiated the JTAG macro, or you use the direct access method and these ports were predefined.

**-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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:	<a href="#">define_dft_dft_configuration_mode</a> on page 699 <a href="#">define_dft_mbist_clock</a> on page 719 <a href="#">define_dft_pmbist_direct_access</a> on page 734 <a href="#">define_dft_test_mode</a> on page 765
Affects these commands:	<a href="#">insert_dft_boundary_scan</a> on page 787 <a href="#">insert_dft_jtag_macro</a> on page 806
Related commands:	<a href="#">read_memory_view</a> on page 875 <a href="#">read_pmbist_interface_files</a> on page 877 <a href="#">write_pmbist_interface_files</a> on page 958 <a href="#">write_pmbist_testbench</a> on page 960
Affected by this attribute:	<a href="#">pmbist_use</a>
Related attributes:	<a href="#">mbist_enable_shared_library_domain_set</a> <a href="#">pmbist_enable_multiple_views</a> <a href="#">pmbist_instruction_set</a> <a href="#">Memory Data Bit Structure Attributes</a> <a href="#">Memory Libcell Attributes</a> <a href="#">Memory Libpin Action Attributes</a> <a href="#">Memory Libpin Alias Attributes</a> <a href="#">Memory Spare Column Attributes</a> <a href="#">Memory Spare Column Map Address Attributes</a> <a href="#">Memory Spare Column Map Data Attributes</a> <a href="#">Memory Spare Row Attributes</a> <a href="#">Memory Spare Row Map Address Attributes</a> <a href="#">Write Mask Bit Attributes</a>

## **insert\_dft ptam**

```
insert_dft ptam
  -power_test_enable {pin|port}
  [-power_test_enable_active {low|high} ]
  [-shift_enable test_signal ]
  [-instruction string] [-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-instruction string`

Specifies the name of the instruction which must be loaded into the IEEE 1149.x TAP controller instruction register to access the PTAM test data register.

`-power_test_enable {pin | port}`

Identifies the power-test-enable pin or port. Asserting this pin to an active state will enable the PTAM logic to override the design's power manager control pins. This serves as a master mode control signal.

**Note:** This cannot be the same pin or port identified by `define_dft test_mode` that control pin sharing if the PTAM I/O are shared

`-power_test_enable_active {high | low}`

Specifies the active value for the power-test-enable pin or port.

*Default:* high

`-preview`

Shows the potential changes without making any modifications to the netlist.

`-shift_enable test_signal`

Designates a shift-enable test signal used to override the PTAM gating logic of the power manager output control signals during scan test.

If you omit this option, the default shift-enable signal specified using `define_dft shift_enable` is used as selected by its `default_shift_enable` attribute.

## Examples

- The following command inserts the PTAM logic into design `chip_top`.

```
insert dft ptam -instruction PTAM -power_test_enable /chip_top/PwrTe \
-directory ${workdir}/testmode_data
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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 1030
- [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*

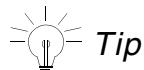
## **insert\_dft rrfa\_test\_points**

```
insert_dft rrfa_test_points
  { -input_tp_file file
    | [-atpg [-atpg_effort {low|medium|high}]
      [-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](#).



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_effort {low | medium | high}`  
Specifies the effort to be used for the ATPG-based testability analysis.

*Default:* low

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

	<p><b>Note:</b> For more information on these options, refer to the <code>build_testmode</code> command in the <i>Command Line Reference</i> (of the Encounter Test documentation).</p>
<code>-control_only</code>	Specifies to insert only control test points.
<code>design</code>	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.
<code>-directory string</code>	Specifies the working directory for Encounter Test.  <b>Note:</b> This option is only required when you run an RRFA-based analysis
<code>-et_log file</code>	Specifies the name of the Encounter Test log file. This file will be generated in the specified directory.  <i>Default:</i> <code>eta_from_rc.log</code>
<code>-gate_clock</code>	Enables the insertion of combinational logic to clock gate the test clocks of the inserted test points.  <b>Note:</b> If the hierarchy in which the observe points will be inserted already contains integrated clock-gating cells, the tool can use the <code>ck_out</code> 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 <code>lp_clock_gating_test_signal</code> 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.  <b>Note:</b> To use this command option you need an Encounter Test license. For more information on the exact product requirements, refer to <a href="#">Encounter Test Product Requirements for Advanced Features</a> in <i>Design for Test in Encounter RTL Compiler</i> .
<code>-gate_clock_test_control test_signal</code>	Specifies the test signal to be used to control the gating logic of the clock of the test point.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 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 ./ip_blocks" ...
```

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

```
insert_dft rrfa_test_points -output_tp_file myfile
```

- The following example inserts 10 test points from the specified test point file `myfile`.

```
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.

```
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.

```
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 765  
[define\\_dft\\_test\\_clock](#) on page 761

# Command Reference for Encounter RTL Compiler

## Design for Test

## **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-max_number_of_testpoints integer`

Specifies the maximum number of test points to be inserted. Selection is based on the specified number of identified test points and the weight of the point resulting from logic cone analysis. The weight indicates the probability of a higher power level if toggling occurs.

`-min_slack integer` Limits the insertion of a test point to those nodes that have the specified minimum slack (in ps).

*Default:* 2000

`-output_tp_file file`

Specifies the name of the generated output file containing the recommended test points.

If you do not specify this option, the test point locations are written to the following file in the working directory:

`TB/testresults/TestPointInsertion.ASSUMESCAN.expt`

`-power_mode mode` Specifies the power mode for which the test power must be reported.

Specify this option with the `-report_virtual_scan_power` option when the design has multiple power modes.

`-preview` Reports the test points to be added, without modifying the design.

`-report_virtual_scan_power "clockFreq [flopTogglePercent]"`

Performs a *virtual* insertion of the test points into the design and measures their impact on the reduction of scan power. Use a floating value to specify the test clock frequency (in MHZ) and the flop-toggle percentage to use in the measurement. If the flop toggle percentage is not specified, it will default to 50% of the test clock frequency.

**Note:** This option is only valid with the `-preview` option.

`-test_control test_signal`

Specifies the test signal to enable the test point. This option is not required when the command is run with the `-preview` option.

**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 765

## **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-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.
<i>design</i>	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 <i>pins</i>	Prevents the specified pins from being considered for shadow logic insertion.
-exclude_shadow_logic { <i>instance</i> ...}	Excludes automatic shadow logic insertion for the specified instances. You can only specify instances of blackboxes or timing models.
-group <i>pins</i>	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:  { <i>input</i> <sub>i</sub> ... <i>output</i> <sub>j</sub> ...}
	Separate the pins by spaces. If you have more than one group, you must specify multiple -group options.
	<b>Note:</b> 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 <i>integer</i>	Limits shadow logic insertion to logic abstract or timing models that have more pins (inputs and outputs) than the specified value.  <i>Default:</i> 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

	<code>no_share</code>	Inserts one scannable observation test point per input and one scannable control test point per output.
	<code>share</code>	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).
<code>-only pins</code>		Restricts the pins to be considered for shadow logic insertion to the specified ones.
<code>-preview</code>		Shows the potential changes, without making any modifications to the netlist.
<code>[-rise -fall]</code>		Specifies the edge of the test clock that is active during test mode operation. These options are only valid in conjunction with <code>-test_clock_pin</code> .
		<i>Default:</i> <code>-rise</code>
<code>-test_clock_pin {port   pin}</code>		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.
<code>-test_control test_signal</code>		Specifies the test signal to use to control DFT logic (the multiplexers after the controlling points).
		<b>Note:</b> You must have specified the test signal using either the <code>define_dft shift_enable</code> or <code>define_dft test_mode</code> 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));
...
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- Using the following examples, bypass logic is used to make the two inputs observable and the three outputs controllable. The first command pairs input i1 to output o1, and input i2 to output o3 (skipping o2). The second command pairs input i1 and output o2.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -mode bypass \
-exclude o2
insert_dft shadow_logic -around U1 -test_control my_TM -mode bypass \
-only {i1 o2}
```

- The following example uses scannable test points and shares these test points as control and observation points.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock_pin CK \
-mode share
```

- The following example uses scannable test points but does not share these test points for control and observation points.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock CK \
-mode no_share
```

- The following example uses scannable test points, shares these test points for control and observation points, and controls by grouping which pins share a common test point. More specifically, i1 and o2 share a test point, and i2 and o1. In addition, no control point is inserted for the net driven by U1/o3.

```
define_dft test_mode -name my_TM -active high TM
insert_dft shadow_logic -around U1 -test_control my_TM -test_clock CK \
-exclude o3 -mode share -group {i1 o2} -group {i2 o1}
```

- The following example automatically inserts shadow logic around all logic abstract and timing model instances.

```
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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 884

Related constraints: [define\\_dft\\_shift\\_enable](#) on page 751

[define\\_dft\\_test\\_clock](#) on page 761

[define\\_dft\\_test\\_mode](#) on page 765

## **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 <i>design</i>	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 <i>integer</i>	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 <i>test_signal</i>	Specifies the controlling shift enable signal.
-write_to_log_file <i>file</i>	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 853.

**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`.

# Command Reference for Encounter RTL Compiler Design for Test

**Note:** You must use the same clock edge when inserting a control flip-flop and an observation flip-flop.

*Default:* -rise

-test\_clock\_pin {*port* | *pin*}

Specifies the test clock that drives the clock pin of the inserted flip-flops during test mode operation. You can specify a port or pin that drives the test clock.

This option is required for all type point types set using the -type option except those of type control\_0 or control\_1.

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

control.observe_0	Inserts a control and an observation point. The control point is forced to the value 0.
control.observe_1	Inserts a control and an observation point. The control point is forced to the value 1.
control.observe_node	Inserts a control and an observation point. The control point is forced to the value of the node specified by -node.
control.scan	Inserts a flip-flop to force a particular value at the specified location during test mode operation. The flip-flop must be remapped to a scan flop before connecting it to a scan chain later on.  <b>Note:</b> This option requires you to specify the <code>-test_clock_pin</code> option.
observe.scan	Inserts a flip-flop to observe the specified location. The flip-flop must be remapped to a scan flip-flop before connecting it to a scan chain later on.  This option requires you to specify the <code>-test_clock_pin</code> option.
scan	Inserts a scannable control and observation test point.  <b>Note:</b> This option requires you to specify the <code>-test_clock_pin</code> option.
sync_0	Inserts a synchronous control test point that forces the control point to the value 0.
sync_1	Inserts a synchronous control test point that forces the control point to the value 1.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

sync_any	Inserts a synchronous control test point that forces the control point to take either the original value or the inverted value.
----------	---

## Examples

- The following example inserts a scannable observation test point, using CLK to drive:

```
insert_dft test_point -location X/out -test_clock_pin CLK -type observe_scan
```

- The following example inserts a control-1 and scannable observation point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type control_observe_1
```

- The following example inserts a scannable control point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type control_scan
```

- The following example inserts a scannable control and observation test point:

```
insert_dft test_point -location X/out -test_control TM \
-test_clock_pin CLK -type scan
```

- The following example inserts an async control-0 test point at hierarchical pin X/out:

```
insert_dft test_point -location X/out -test_control TM -type async_0 \
-test_clock_pin CK -fall.
```

- The following example inserts a synchronous control test point that forces the control point to the value 1 at hierarchical pin X/out:

```
insert_dft test_point -location X/out -test_control TM -type sync_1 \
-test_clock_pin CK -fall.
```

- The following example inserts two observation test points, one for pin1, pin2 and pin3, and the other for pin4, pin5 and pin6.

```
insert_dft test_point -type observe_scan -max_observe_share 3 \
-location pin1 pin2 pin3 pin4 pin5 pin6
```

## 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 379

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

Related constraints: [define\\_dft\\_shift\\_enable](#) on page 751

[define\\_dft\\_test\\_clock](#) on page 761

[define\\_dft\\_test\\_mode](#) on page 765

Related attributes: [Test Clock Attributes](#)

[Test Signal Attributes](#)

## **insert\_dft user\_test\_point**

```
insert_dft user_test_point -location {pin|port|subport}
  -cell {design|subdesign|libcell}
  {-cfi {pin|port} | -no_cfi} [-cfo {pin|port}]
  -connect string [-connect string]...
  -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.

### **Example**

- 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}
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 379

## **insert\_dft wrapper\_cell**

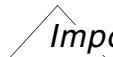
```
insert_dft wrapper_cell -location pin_list
  [-floating_location_ok]
  [-skipped_locations_variable Tcl_variable]
  [-shared_through {buffer|inverters_and_buffers|combinational}
    [-no_mux_before_shared_cell]]
  [-wck pin] -wsen_in pin -wsen_out pin
  { -decoded_select_cfi pin
    | -wint {pin|port|constant}
      -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`



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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

	<p>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.</p>
<code>-design design</code>	Specifies the design in which you want to insert the wrapper cell. This option is required if multiple designs are loaded.
<code>-dont_reuse_input_wrappers_for_output_ports</code>	Allows to insert dedicated wrapper cells in the fanin of output ports that are being fed by wrapper cells previously inserted for input ports.
<code>-exclude pin_list</code>	Specifies the list of pins or ports to be excluded from wrapper insertion.
<code>-exclude_comb_feedthrough_paths</code>	Specifies whether to exclude pins or ports from wrapper insertion if combinational logic is found on the path from input to output.
<code>-floating_location_ok</code>	Specifies to insert a wrapper cell even if the specified pin (location) is floating.
<code>-guard {0   1}</code>	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.
<code>-input_shared_threshold integer</code>	Specifies the maximum number of scannable flops that can be shared in wrapper cells. When the number of scannable flops exceeds the specified threshold value, a single dedicated wrapper cell will be inserted for the input port.  <i>Default:</i> 10
<code>-inside_core</code>	Inserts the wrapper cells associated with hierarchical pins inside the core module.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-location pin_list`

Specifies one or more pins that identify where the wrapper cell must be inserted.

**Note:** When specifying the pins of a blackbox instance use the RC pin name to identify the input or output pins.

`-map_to_mux_for_dedicated_wrapper`

Specifies to map the mux logic (associated with the CFI to CFO path and the optional Guard control path in the dedicated wrapper cell) to binary muxes from the target library. The binary muxes will be marked preserved (`preserve` attribute on libcell is set to `true`) for optimization. The instance names given to the muxes inserted for the CFI to CFO path and Guard control path are `dwc_cfi_mux` and `dwc_guard_mux` respectively.

**Note:** If you omit this option, the technology mapper will select the appropriate cell type(s) which satisfy the timing constraints.

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

`-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.

`-wcap driver`

Specifies the capture control source pin or port name.

**Note:** This option is ignored if you specified the `-decoded_select_cfi` option.

`-wck driver`

Specifies the test clock source pin or port name. If specified, this pin will connect to the clock pin of the inserted dedicated wrapper cells.

**Note:** If this option is not specified, the tool will identify a local test clock to be used to connect to the clock pin of the inserted dedicated wrapper cells.

`-wig driver`

Specifies the in-guard control source pin or port name for an inward-facing wrapper cell.

**Note:** This signal will typically be controlled by the `-wext` control signal.

`-wint {pin|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 `-decoded_select_cfi` option.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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 <i>test_signal</i>	Specifies the test signal used as capture signal for the Wrapper Instruction Register.
-design <i>design</i>	Specifies the design in which you want to insert the wrapper instruction register. This option is required if multiple designs are loaded.
-prefix <i>string</i>	Specifies the string to be prepended to the WIR name.
-selectwir <i>test_signal</i>	Specifies the test signal used as select signal for the Wrapper Instruction Register.
-shiftwr <i>test_signal</i>	Specifies the name of the shift signal for the Wrapper Instruction Register.
-updatewr <i>test_signal</i>	Specifies the test signal used as update signal for the Wrapper Instruction Register.
-wrck { <i>pin port</i> }	Specifies the pin or port on which the Wrapper Instruction Register clock is defined.

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

`-wrstn test_signal`

Specifies the test signal used as reset signal for the Wrapper Instruction Register.

`-wsi port`

Specifies the port used as Wrapper Serial Input.

`-wso port`

Specifies the port used as Wrapper Serial Output.

## **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]
[-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_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.

```
define_dft dft_configuration_mode -name functional -type wrapper -usage mission \
    -mode_enable_low TS1 TS2 TS3 TS4
define_dft dft_configuration_mode -name serialIntest -type wrapper -usage intest \
    -mode_enable_low TS4 TS3 TS2 -mode_enable_high TS1
define_dft dft_configuration_mode -name serialExtest -type wrapper -usage extest \
    -mode_enable_low TS4 TS3 TS1 -mode_enable_high TS2
define_dft dft_configuration_mode -name parallelIntest -type wrapper \
    -usage intest -mode_enable_low TS4 TS3 -mode_enable_high TS2 TS1
define_dft dft_configuration_mode -name parallelExtest -type wrapper \
    -usage extest -mode_enable_low TS4 TS2 TS1 -mode_enable_high TS3
define_dft dft_configuration_mode -name parallelCompress -type wrapper \
    -usage intest -mode_enable_low TS4 TS2 -mode_enable_high TS3 TS1
insert_dft wrapper_mode_decode_block \
    -create_wrapper_shift_enables_for_delay_test DFTWSE
```

## Related Information

[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
  [-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...
    [-use_wir_macro instance] [-bypass_reg]
  [-auto_create] [-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**

**-auto\_create** Automatically creates the necessary compression control signals as top-level ports.

For the names of the auto-created ports, refer to the "Hierarchical Test" chapter.

**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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 Coret 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 817

## **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 379

Related commands:      [write\\_dft\\_abstract\\_model](#) on page 914

[write\\_hdl](#) on page 278 (-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 787

[write\\_io\\_speclist](#) on page 948

## **read\_memory\_view**

```
read_memory_view
  {-config_file_view config_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**

`-config_file_view 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 825

Related attributes:  
[Memory Data Bit Structure Attributes](#)  
[Memory Libcell Attributes](#)  
[Memory Libpin Action Attributes](#)  
[Memory Libpin Alias Attributes](#)

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

[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 825

Related commands: [write\\_pmbist\\_interface\\_files](#) on page 958

## **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 890  
[set\\_opcg\\_equivalent](#) on page 894

## 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 BTI 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 781

identify test mode registers on page 781

set scan equivalent on page 896

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**

<i>libcell</i>	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 878

Related command: [set\\_opcg\\_equivalent](#) on page 894

## **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**

<i>libcell</i>	Specifies a non-scan library cell to be removed from the scan-equivalency table.
----------------	--

### **Example**

The following example removes the `snl_ffqx1` cell from the scan-equivalency table.

```
reset_scan_equivalent snl_ffqx1
```

### **Related Information**

Affects this command: [replace\\_scan](#) on page 880

Related command: [set\\_scan\\_equivalent](#) on page 896

## **set\_compatible\_test\_clocks**

```
set_compatible_test_clocks
  {-all | -none | list_of_test_clocks}
  [-dont_check_dft_rules] [-design design]
```

Specifies the compatible test clocks whose related scan flip-flops can be merged into a single scan chain using lockup elements in between. By default, no test clocks are assumed compatible unless they are defined as independent test clocks in the same test clock domain.

**Note:** This command applies only to the muxed scan style.

Test clocks that are declared compatible belong to the same DFT clock domain.



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**

<code>-all</code>	Specifies that all test clocks are compatible.
<code>-design <i>design</i></code>	Specifies the design for which you want to specify compatible test clocks.
<code>-dont_check_dft_rules</code>	Prevents the DFT rules from being checked automatically after specifying the compatible test clocks.
<code><i>list_of_test_clocks</i></code>	Specifies the compatible test clocks. You must specify the test clock object name.  <b>Note:</b> 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 <code>dft_mix_clock_edges_in_scan_chains</code> root attribute to true. By default, only same edge clocks are mixed.
<code>-none</code>	Specifies that none of the test clocks are compatible.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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**

<code>-edge_mode_pin pin</code>	Specifies the edge mode pin of the OPCG cell to connect to.
<code>-loop_back_pin pin</code>	Specifies the loopback pin of the OPCG cell to connect to.
<code>-opcg_cell libcell</code>	Specifies the OPCG library cell to map to.
<code>-pin_map list_of_pin_groups</code>	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 <i>list_of_pin_groups</i> has the following format:  <code>{ {scan_pin opcg_pin} {scan_pin opcg_pin} ... }</code>
<code>-scan_cell libcell</code>	Specifies a scan flip-flop library cell to be replaced.
<code>-tieoff_pins string</code>	Specifies the tie-off value for extra pins on the OPCG cell.  The <i>string</i> has the following format:  <code>{ {pin tie_off_value} {pin tie_off_value} ... }</code>
	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 SDFFQ_X1M -opcg_cell S2DFFQQN_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 878

Related command: [reset\\_opcg\\_equivalent](#) on page 890

## **set\_scan\_equivalent**

```
set_scan_equivalent  
  -non_scan_cell libcell -scan_cell libcell  
  [-tieoff_pins string] [-pin_map list_of_pin_groups]
```

Controls the scan-equivalent cell type that is used during the conversion of a non-scan flip-flop which passes the DFT rule checks to a scan flop. Use the `replace_scan` command to perform the actual conversion to scan.

**Note:** The RC-DFT engine automatically derives the scan data input, scan data output, and other test signals from the `test_cell` description of the scan flop in the target library.

### **Options and Arguments**

`-non_scan_cell libcell`

Specifies a non-scan flip-flop library cell.

`-pin_map list_of_pin_groups`

Indicates how to map a pin from the non-scan flop to a pin in the scan flop when the pin names in the cells do not match.

The `list_of_pin_groups` has the following format:

```
{ {non_scan_pin scan_pin} {non_scan_pin  
scan_pin} ... }
```

`-scan_cell libcell`

Specifies a scan flip-flop library cell.

`-tieoff_pins string`

Specifies the tie-off value for extra scan cell pins.

The `string` has the following format:

```
{ {pin value} {pin value} ... }
```

The value can be a logic 0 or 1.

### **Example**

The following example assumes that the pin names in the non-scan and scan flip-flops match, and that there are no extra pins in the scan flop to be tied off.

```
set_scan_equivalent -non_scan_cell snl_ffqx1 -scan_cell snl_sffqx1
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

#### 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 880

Related command: [reset\\_scan\\_equivalent](#) on page 891

## **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 *]
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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 802

[insert\\_dft\\_rrfa\\_test\\_points](#) on page 833

[insert\\_dft\\_test\\_point](#) on page 849

[insert\\_dft\\_user\\_test\\_point](#) on page 855

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

-compression	Creates pin-assignment files for full scan mode, compression mode and XOR decompression mode. The following files are generated: <i>topmodulename</i> .FULLSCAN.pinassign, <i>topmodulename</i> .COMPRESSION.pinassign, and <i>topmodulename</i> .COMPRESSION_DECOMP.pinassign.  <b>Note:</b> 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.
<i>design</i>	Specifies the top module for which to write ATPG.
-dft_configuration_mode <i>dft_configuration_mode</i>	Writes ATPG information for the specified scan mode name.  <b>Note:</b> This option is only valid with the -stil option.
<i>file</i>	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.  <b>Note:</b> 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: <ul style="list-style-type: none"><li>■ <i>top_module</i>.testproc</li><li>■ <i>top_module</i>.dofile</li></ul>
-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.  <i>Default:</i> nanoseconds
-stil	Creates an interface file in the IEEE Standard Test Interface Language (STIL) format (IEEE format 1450.1).  <b>Note:</b> The generated STIL format is TetraMAX compatible.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 761

Affected by these attributes:      [Actual Scan Chain Attributes](#)

## **write\_bsdl**

```
write_bsdl
  [-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.bsdl*.

**-directory *string***

Specifies the directory to which the output files must be written.

**-expose\_ports\_with\_pinmap**

Specifies to only expose functional and test ports with package pinmap information to the output BSDL file.

Additionally, this option prevents the writing of other ports connected in the boundary-scan register without package pinmap information. The names of these other ports will not appear in the BOUNDARY\_REGISTER section of the BSDL file. Rather, these port names will be represented with an asterisk (\*) and their associated boundary-scan cells will be represented as INTERNAL as shown in the following BSDL snippet:

```
attribute BOUNDARY_REGISTER of test: entity is
  ...
  "8  (BC_OUT; *, INTERNAL, X)," &
```

When the -expose\_ports\_with\_pinmap is not specified, all ports in the design will be written to each relevant section of the BSDL file, regardless of whether any pinmap information has been provided for any ports.

Package pinmap information may be provided during boundary-scan insertion or when writing the BSDL file using the -pinmap\_file option.

**-include\_private\_instructions**

Specifies to include register access information for private instructions in the BSDL file.

**-pinmap\_file file** Specifies the name of the pinmap file to be used to create a BSDL file.

**Note:** This file can have fewer pin-to-pad bonding requirements than the pinmap file specified for the boundary scan insertion.

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_bsdl -directory . -bsdl_package_name MY_BIDIR_PKG -bsdlout bsdlout
```

This causes the following line to be added to file bsdlout:

```
use MY_BIDIR_PKG.all ;
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **Related Information**

[Writing a BSDL File in Design for Test in Encounter RTL Compiler](#)

Affected by this command: [insert\\_dft\\_boundary\\_scan](#) on page 787

Related constraints: [define\\_dft\\_shift\\_enable](#) on page 751

[define\\_dft\\_test\\_clock](#) on page 761

[define\\_dft\\_test\\_mode](#) on page 765

## **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 (`number_of_chains-1`), while for `wide2` type of masking, the number must equal the specified number of chains minus two (`number_of_chains-2`).

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-bitwise\_compressor** Generates a compressor with bitwise xorring 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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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 opmisrl.v -info_file opmisrl.info
....
```

- The following command generates an asymmetric pipelined XOR-based compression macro, with masking logic of type wide2, with five top-level chains, of which two are shared with the mask control signals, with separate mask ports, with additional logic to bypass the block for compression inserted at the block level, and with two pipeline stages at the scan data output side.

```
write_compression_macro -chains 5 -sub_chains 9 -asymmetric_scan_in 3 \
    -shared_scan_in_pins "CME0 CME1" -mask wide2 -separate_mask_ports \
    -block_select -scan_out_pipeline_depth 2 -file my_comp_macro.v
```

- The following command generates a pipelined XOR-based compression macro, with masking logic of type wide1, with two top-level chains, with one pipeline stage at the scan data input side, six pipeline stages at the scan data output side, and without additional muxing logic.

```
write_compression_macro -chains 2 -sub_chains 5 -mask wide1 \
    -scan_in_pipeline_depth 1 -scan_out_pipeline_depth 6 -no_fullscan_muxing
```

- The following command generates an asymmetric SmartScan-based compression macro with both parallel and serial interface, with masking logic of type wide1, with eight fullscan chains, 32 compression channels and a SmartScan ratio of 4.

```
write_compression_macro -chains 8 -sub_chains 32 -mask wide1 \
    -compressor smartscan_xor -decompressor xor -smartscan_ratio 4
```

- The following command generates an asymmetric SmartScan-based compression macro with serial interface only, with masking logic of type wide1, with two fullscan chains, 32 compression channels and a SmartScan ratio of 4.

```
write_compression_macro -chains 2 -sub_chains 32 -mask wide1 \
    -compressor smartscan_xor -decompressor xor -smartscan_ratio 4 \
    -smartscan_serial_only
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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>
    llatch 5   DFT_Lockup_g1
    bit 6       out2_reg_4 <test_clk2/fall>
    ...
    bit 10      out2_reg_8 <test_clk2/fall>
    llatch 10   DFT_Lockup_g348
    bit 11      out3_reg_4 <test_clk3/fall>
    ...
    ...
```

## Command Reference for Encounter RTL Compiler

### Design for Test

```
bit 18      out1_reg_2 <test_clk1/rise>
bit 19      out1_reg_3 <test_clk1/rise>
llatch 19   DFT_lockup_g349
bit 20      out2_reg_0 <test_clk2/rise>
...
bit 23      out2_reg_3 <test_clk2/rise>
llatch 23   DFT_lockup_g350
bit 24      out3_reg_0 <test_clk3/rise>
...
bit 27      out3_reg_3 <test_clk3/rise>
-----
rc:/> write_dft_abstract_model
scan style is muxed_scan
# writing abstract model for 1 scan chain
define_dft abstract_segment -module test \
  -name test_AutoChain_1 \
  -sdi DFT_sdi_1 -sdo DFT_sdo_1 \
  -shift_enable_port SE -active high \
  -clock_port clk1 -fall \
  -tail_clock_port clk3 -tail_edge_rise \
  -length 27
```

To avoid naming collisions when reading in a scan abstract model, the segment names are prefixed with the module name.

### Related Information

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 871

[write\\_hdl](#) on page 278 (-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**

<i>design</i>	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 <i>directory</i>	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 806](#)

[insert\\_dft\\_mbist on page 817](#)

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]
  | [-compression]
    [-dft_configuration_mode dft_config_mode]...
    [-build_faultmodel_options string]
    [-build_model_options string]
    [-build_testmode_options string]
    [-atpg_options string] [-effort 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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.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

**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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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_OP CGModeName.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_OP CGModeName.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_OP CGModeName`—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_OP CGModeName`—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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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_OP CGModeName.pinassign`
  - ❑ `topModuleName.FULLSCAN_OP CGModeName.seqdef`
- If you also specified the `-compression` option, the following files are also written:
  - ❑ `topmodulename.COMPRESSION_OP CGModeName.pinassign`
  - ❑ `topmodulename.COMPRESSION_OP CGModeName.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.

## Command Reference for Encounter RTL Compiler

### Design for 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, COMRESSION\_DECOMP, OPMISRPLUS,  
OPMISRPLUS\_DECOMP, FULLSCAN

**Note:** If specified, the FULLSCAN compression mode must be specified last.

- In the default flow, you cannot combine this option with the `-delay` option.
- In the true time flow, you can combine this option with the `-delay` option, but you can only specify two configuration modes because the true time flow can only handle two modes.

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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`

`-effort {low | medium | high}`

Specifies the ATPG effort level to expend in resolving faults. Increasing effort will generally result in resolving more faults, but will require more processing time, sometimes significantly more.

*Default:* low

**Note:** This option does not apply to the true time flow.

`-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:

```
./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" ...
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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_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.

`-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.

## 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
```

## 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.



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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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*

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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](#)

## **write\_et\_dfa**

```
write_et_dfa
  [-library string]
  [-effort 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*

**-effort {low | medium | high}**

Specifies the ATPG effort level to expend in resolving faults. Increasing effort will generally result in resolving more faults, but will require more processing time, sometimes significantly more.

*Default:* low

**-include\_redundant\_faults**

Specifies to include the redundant faults during DFA analysis.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

```
write_et_dfa -directory dfa -library $sim/mylib.v
```

Examining the `dfa` directory that was generated shows the following files:

```
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 802

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 LBIST testmode

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `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 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

`-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_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.

## Command Reference for Encounter RTL Compiler

### Design for 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.

`-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 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 811

[write\\_logic\\_bist\\_macro](#) on page 950

## **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**

<code>-bsdl file</code>	Specifies the name of the BSDL file to be used for the boundary scan verification.
-------------------------	--

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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/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"
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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
```

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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  
    [-min slack integer] [design]
```

Generates a `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 `notpfile` keyword of the `analyze_random_resistance` and `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 `dft_dont_scan` attribute set to `true`
  - The subdesign or instance has the `preserve` attribute set to `true`, `1`, or `size_ok`
  - The instance has the `hard_region` attribute set to `true`
  - The subdesign or instance has been scan abstracted (using either native or CTL model)

The pins that are included in the file satisfy any of the following conditions:

- The pin has a slack less than the min\_slack
  - The pin has the preserve attribute set to true, 1, or size ok

In addition, any net that does not have its `preserve` attribute set to `false`, is written to the file.

If a subdesign is already written into the file, then the instances, pins or nets in the subdesign are not written out again. Similarly, if the instance is included in the file, then its pins and nets are not written out to the file.

## Options and Arguments

*design* Specifies the design on which to perform test point selection...

If you omit the design name, the top-level design of the current directory of the design hierarchy is used.

-min slack *integer*

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] [-effort string] [-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](#).

# Command Reference for Encounter RTL Compiler

## Design for Test

## 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

-effort {low | medium | high}

Specifies the ATPG effort level to expend in resolving faults. Increasing effort will generally result in resolving more faults, but will require more processing time, sometimes significantly more.

*Default:* low

- 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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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:

```
./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:

```
write_et_rrfa -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_rrfa -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.

`-rrfa_options string`

Specifies the extra options to run RRFA-based testability analysis in a string.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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:

```
rc:/> shell ls rrfra
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.

## **Command Reference for Encounter RTL Compiler**

### Design for Test

---

#### **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 787

[insert\\_dft\\_mbist](#) on page 817

[insert\\_dft\\_ptam](#) on page 830

Related command:      [read\\_io\\_speclist](#) on page 874

## **write\_logic\_bist\_macro**

```
write_logic_bist_macro
    -channels integer -max_length_of_channels integer
    [-clocks integer]
    [-add_rumbist_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]
    [-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_rumbist_support`

Inserts the LBIST macro with JTAG RUNBIST support.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

**-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.

**-reset\_patterns *integer***  
Specifies the default number of reset test patterns to be executed.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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 811

[write\\_et\\_lbist](#) on page 934

## **write\_mbist\_testbench**

```
write_mbist_testbench
    [-create_embedded_test_options string]
    [-testbench_directory string]
    [-ncsim_library file_list]
    [-directory string] [-script_only] [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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `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*

`-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:

## Command Reference for Encounter RTL Compiler

### Design for Test

```
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.

`-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
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- 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 787  
[insert dft jtag macro](#) on page 806  
[insert dft mbist](#) on page 817

Related commands:

[write dft rtl model](#) on page 917  
[write et bsv](#) on page 926  
[write et mbist](#) on page 938  
[write hdl](#) on page 278

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 958](#)

Related command:      [read\\_pmbist\\_interface\\_files on page 877](#)

## **write\_pmbist\_testbench**

```
write_pmbist_testbench
    [-create_embedded_test_options string]
    [-testbench_directory string]
    [-ncsim_library file_list]
    [-directory string] [-script_only] [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

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- `irun.simsimscript.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*

`-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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

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.

`-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.

*Default: current\_working\_directory/  
mbist\_testbench*

## Examples

- In the following example a single PMBIST instruction set was implemented in the design:

```
write_pmbist_testbench -testbench_directory ./mbist_verilog_testbenches \  
-ncsim_library ../$simulation_verilog_libraries \  
-directory ./mbist_verification_scripts
```

## Command Reference for Encounter RTL Compiler

### Design for Test

---

- The following example assumes you are working in a multiple block flow (PMBIST was inserted on multiple designs or blocks) using separate JTAG instructions to access each block's MBIST engines. When merged into the TOP level, each instruction set is labeled with a unique integer suffix by default.

```
write_pmbist_testbench \
    -create_embedded_test_options \
    interfacefilelist="TOP_1_pattern_control.txt, \
    TOP_1_mbisttpn_tdr_map.txt, TOP_1_mbistsch_tdr_map.txt \
    TOP_1_mbistchk_tdr_map.txt, TOP_1_test_def.txt" \
    -testbench_directory ./mbist_verilog_testbenches \
    -ncsim_library../simulation_verilog_Libraries \
    -directory ./mbist_verification_scripts
```

### Related Information

Affected by these commands: [define\\_dft\\_pmbist\\_direct\\_access](#) on page 734

[insert\\_dft\\_boundary\\_scan](#) on page 787

[insert\\_dft\\_jtag\\_macro](#) on page 806

[insert\\_dft\\_pmbist](#) on page 825

Related commands: [read\\_hdl](#) on page 218

[write\\_pmbist\\_interface\\_files](#) on page 958

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_configuration_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**

<i>-chains chain</i>	Specifies the scan chains that must be grouped in the same partition by the physical design tool. Use the <a href="#"><u>report_dft_chains</u></a> 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.  <b>Note:</b> Requires version 5.5.
<i>design</i>	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.
<i>-dft_configuration_mode dft_configuration_mode_name</i>	Specifies the configuration mode for which to write the scan definition
<i>-dont_split_by_library_domains</i>	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.

## Command Reference for Encounter RTL Compiler

### Design for Test

---

`-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:

```
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 969
- [clock\\_gating](#) on page 971
- [clock\\_gating\\_connect\\_test](#) on page 973
- [clock\\_gating\\_declone](#) on page 974
- [clock\\_gating\\_import](#) on page 975
- [clock\\_gating\\_insert\\_in\\_netlist](#) on page 977
- [clock\\_gating\\_insert\\_obs](#) on page 978
- [clock\\_gating\\_join](#) on page 980
- [clock\\_gating\\_remove](#) on page 982
- [clock\\_gating\\_share](#) on page 984
- [clock\\_gating\\_split](#) on page 986
- [read\\_saif](#) on page 988
- [read\\_tcf](#) on page 993
- [read\\_vcd](#) on page 998
- [report\\_clock\\_gating](#) on page 1002
- [report\\_operand\\_isolation](#) on page 1003
- [report\\_power](#) on page 1004
- [state\\_retention](#) on page 1005
- [state\\_retention\\_connect\\_power\\_gating\\_pins](#) on page 1006
- [state\\_retention\\_swap](#) on page 1007
- [write\\_forward\\_saif](#) on page 1008

## **Command Reference for Encounter RTL Compiler**

### Low Power Synthesis

---

- [write\\_saif](#) on page 1010
- [write\\_tcf](#) on page 1012

## **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**

<code>-clean_up_netlist</code>	Requests to remove unreachable logic in the netlist as this can affect the accuracy of the estimation.  <b>Note:</b> Unreachable logic is removed from the input netlist, without changing the logic functionality.
<code>-design <i>design</i></code>	Specifies the design for which to build the power models.
<code>-relative <i>instance_list</i></code>	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.
```

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

- 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 1004

Affected by this attribute: [lp\\_insert\\_clock\\_gating](#)

## **clock\_gating**

```
clock_gating
{ connect_test | declone | import | insert_in_netlist
| insert_obs | join | remove | share | split}
```

Manipulates a netlist for clock gating.



The `clock_gating` commands only work on a mapped netlist.

### **Options and Arguments**

<code>connect_test</code>	Connects the test input of all clock-gating logic.
<code>declone</code>	Merges clock-gating instances driven by the same inputs.
<code>import</code>	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.
<code>insert_in_netlist</code>	Inserts clock-gating logic on a mapped netlist.
<code>insert_obs</code>	Inserts and connects observability logic.
<code>join</code>	Joins multiple-stage clock-gating logic into a single clock-gating instance with a complex enable function.
<code>remove</code>	Removes the specified clock-gating logic.
<code>share</code>	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.
<code>split</code>	Splits a single clock-gating instance with a complex enable function into multiple stages of clock-gating logic.

### **Related Information**

Related commands:	<a href="#"><u>clock_gating connect test</u></a> on page 973
	<a href="#"><u>clock_gating declone</u></a> on page 974
	<a href="#"><u>clock_gating import</u></a> on page 975

## **Command Reference for Encounter RTL Compiler**

### Low Power Synthesis

---

[clock gating insert in netlist](#) on page 977

[clock gating insert obs](#) on page 978

[clock gating join](#) on page 980

[clock gating remove](#) on page 982

[clock gating share](#) on page 984

[clock gating split](#) on page 986

## **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 1002

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 1002

Affected by this attribute: [lp\\_clock\\_gating\\_max\\_flops](#)

## **clock\_gating import**

```
clock_gating import  
  [-start_from instance] [-hierarchical]  
  [-detail [-verbose]]
```

Processes clock-gating instances that were either manually inserted or inserted by third-party tools to make them recognizable as clock-gating instances by the RC-LP engine.

The command returns the total number of instances imported.

Currently, the RC-LP engine recognizes the following structures as clock-gating instance:

- Two-input AND or NAND gates that have a clock signal driving one of the inputs
  - In this case, the other pin is assumed to be the enable pin.  
**Note:** If the other pin is part of the test network, the 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 <i>instance</i>	Starts processing clock-gating instances that were not inserted by RTL Compiler from the specified hierarchical instance.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

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 1002

## **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 1002

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**

<code>-disable_clock</code>	Specifies to gate the clock of the observability flip-flops.  <b>Note:</b> One gating cell is inserted per flip-flop. The RC-LP engine creates a separate subdesign for each gating cell.
<code>-exclude instance</code>	Prevents insertion of observability logic in the specified hierarchical instances.
<code>-hierarchical</code>	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.
<code>-ignore_clock_constraint</code>	Inserts observability logic based on physical connectivity.  By default, observability logic is inserted based on clock constraints defined with the <code>define_clock</code> command.
<code>-libcell libcell</code>	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 320

[report\\_clock\\_gating](#) on page 1002

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**

<code>-hierarchical</code>	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.
<code>-max_level integer</code>	Specifies the maximum levels of clock-gating instances that can be combined. If you specify n, n+1 stages can be combined.  <i>Default:</i> 1 allowing 2 levels to be combined.
<code>-multi_fanouts</code>	Allows joining even if the root stage clock-gating instance is driving multiple clock-gating instances.
<code>-start_from instance</code>	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`

## **Command Reference for Encounter RTL Compiler**

### Low Power Synthesis

---

#### **Related Information**

[Consolidating Multi-Stage Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Related command: [report clock gating](#) on page 1002

## **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.

By default, this command only affects the current level of the hierarchy.

- |             |                                       |
|-------------|---------------------------------------|
| -no_verbose | Suppresses info and warning messages. |
| -obs_only   | Removes only the observability logic. |

### **Example**

- The following command removes all clock-gating instances in the hierarchy of subdesign sub1.

```
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 1002

## **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 1002

## **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]
```

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

- The following command allows splitting a single clock-gating instance with a complex enable function into three stages of clock-gating instances across the hierarchy starting from the current directory.

```
clock_gating split -hierarchical -max_level 2
```

#### Related Information

[Splitting Clock-Gating Instances into Multiple Levels of Clock-Gating Logic in Low Power in Encounter RTL Compiler](#)

Related command: [report clock\\_gating](#) on page 1002

**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:

```

prob_new = (prob_old + w * prob_spec) / (1+w)
tr_new = (tr_old + w * tc_spec / duration_spec) / (1+w)

```

where `prob_old` and `tr_old` are the stored values, and `prob_spec`, `tc_spec`, and `duration_spec` are the probably, toggle count, and duration values derived from the new SAIF file.

# Options and Arguments

*file*              Specifies the name of the SAIF file. The file can have any name, suffix, or length.

**-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.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

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.

`-scale scale_factor`

Scales the toggle counts in the SAIF file by dividing them by the specified factor. Use a positive (non-zero) floating number.

*Default:* 1.0

`-update`

Indicates that you are updating the probability values and toggle counts.

`-verbose`

Prints a message for each net that is asserted.

*Default:* Silent mode. Prints the percent completion messages.

`-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

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

## Examples

For the following examples, consider the following SAIF file (`and2.saif`):

```
(SAIFFILE
(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%)
```

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

- 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:

```
(SAIFILE  
...  
(TIMESCALE 1 ns)  
(DURATION 1000.00)  
(INSTANCE a  
  (NET  
    (in1  
      (T0 900) (T1 100) (TC 5)  
    )  
  )  
)
```

The following command updates the stored switching activities with the data in the `and2_new.saif` and gives a two times higher weight on the values in the `and2_new.saif` file.

```
read_saif -update -weight 2 and2_new.saif
```

Check the asserted toggle rates on nets in1:

```
rc:/> get_attr lp_asserted_toggle_rate nets/in1  
0.006333
```

This can be calculated as follows:

$$\begin{aligned} tr_{\text{new}} &= (tr_{\text{old}} + w * tc_{\text{spec}} / duration_{\text{new}}) / (1+w) \\ &= (0.009000 + 2 * 0.005000) / (1+2) = 0.006333 \end{aligned}$$

### 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 1004

Related command: [write\\_saif](#) on page 1010

Sets these attributes: [lp\\_asserted\\_probability](#)

## **Command Reference for Encounter RTL Compiler**

### Low Power Synthesis

---

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:

$$\begin{aligned} \text{prob\_new} &= (\text{prob\_old} + w * \text{prob\_spec}) / (1+w) \\ \text{tr\_new} &= (\text{tr\_old} + w * \text{tc\_spec}/\text{duration\_spec}) / (1+w) \end{aligned}$$

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.



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**

<code>file</code>	Specifies the name of the TCF file. The file can have any name, suffix, or length.
<code>-ignorecase</code>	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.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

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`.

*Default:* 1.0

## Examples

- Consider the following TCF file (example1.tcf):

```
tcffile () {
    tcfversion      :      "1.0";
    duration        :      "1.500000e+05";
    unit            :      "ns";
    instance () {
        pin () {
            "i_12/z"          :      "0.566 747";
            "n_n1/B"          :      "0.516 475";
            "hier1/i_0/z"     :      "0.5 500";
            "hier1/n_n0/z"   :      "0.5 500";
            "hier1/n_n0/A"   :      "0.5 500";
            "hier1/i_0/A"     :      "0.5 500";
            "hier1/i_0/B"     :      "0.5 500";
            "n_n1/A"          :      "0.61 516";
        }
    }
}
```

To read this TCF file (example1.tcf), use the following command:

```
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        :      "0.75000e+05";
    unit            :      "ns";
    instance () {
        pin () {
            "i_12/z"          :      "0.566 747";
            "n_n1/B"          :      "0.516 475";
            "hier1/i_0/z"     :      "0.5 500";
            "hier1/n_n0/z"   :      "0.5 500";
            "hier1/n_n0/A"   :      "0.5 500";
            "hier1/i_0/A"     :      "0.5 500";
            "hier1/i_0/B"     :      "0.5 500";
            "n_n1/A"          :      "0.61 516";
        }
    }
}
```

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
```

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

- Consider the following TCF file (example1.tcf):

```
tcffile () {
    tcfversion : "1.0";
    duration   : "1.000000e+05";
    unit       : "ns";
    instance () {
        pin () {
            "i_0/A"      : "0.5 500";
            "i_0/B"      : "0.6 600";
            "i_0/Z"      : "0.7 700";
        }
    }
}
```

Assume you read this TCF file with the following command:

```
rc:> read_tcf example1.tcf
```

Now consider the following TCF file (example2.tcf):

```
tcffile () {
    tcfversion : "1.0";
    duration   : "1.500000e+05";
    unit       : "ns";
    instance () {
        pin () {
            "i_0/A"      : "0.5 600";
            "i_0/B"      : "0.6 750";
            "i_0/Z"      : "0.7 900";
        }
    }
}
```

Assume you read this TCF file with the following command:

```
rc:> read_tcf -update -weight 0.5 example2.tcf
```

You would get the same result by:

- a. Creating the following TCF file (example3.tcf)

```
tcffile () {
    tcfversion : "1.0";
    duration   : "1.500000e+05";
    unit       : "ns";
    instance () {
        pin () {
            "i_0/A"      : "0.5 700";
            "i_0/B"      : "0.6 850";
            "i_0/Z"      : "0.7 1000";
        }
    }
}
```

- b. Reading the example3.tcf file using the following command:

```
read_tcf example3.tcf
```

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

#### 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 1004

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.



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.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

`-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.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

`-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 -vcf_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 1004

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 operand\_isolation**

Refer to [report ple](#) 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 1006

[state\\_retention swap](#) on page 1007

## **state\_retention connect\_power\_gating\_pins**

```
state_retention connect_power_gating_pins
```

Connects the power gating pins according to the driver specifications.

Use this command if you

- Replaced the sequential cells with state-retention cells after mapping and did not specify to hook up the power gating pins at that time (used `state_retention swap` command without the `-connect_power_gating_pins` option).
- Specified the driver specifications (`state_retention define_driver` commands) after mapping (although the mapping instructions were given before mapping)

### **Related Information**

[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_srpq_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](#)

# Command Reference for Encounter RTL Compiler

## Low Power Synthesis

**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/cq > my.saif
```

## **Command Reference for Encounter RTL Compiler**

### **Low Power Synthesis**

---

- 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 1029  
                          [read\\_saif](#) on page 988

## **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**

<code>-boundary_only</code>	Writes out the switching activities of the primary inputs ports and the leaf sequential instance output pins.
<code>-computed</code>	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.
<code>-duration <i>simulation_period</i></code>	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 <code>3e-3/ns</code> , the default printed toggle count is 300000. If the simulation period is set to <code>1e+5 ns</code> , the printed toggle count will be 300.  <b>Note:</b> Do not choose the duration too small, otherwise the toggle count will be rounded to 0, because only integer numbers are written out.  <i>Default:</i> <code>1e+9 ns</code>
<code><i>file</i></code>	Specifies the name of the file to which to write the probability values and toggle count values.
<code>-include_hier_ports</code>	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 988

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

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

#### **write\_tcf**

```
write_tcf [-duration simulation_period] [-computed]
           [-hierarchical] [-include_hier_ports] [-pin] [> file]
```

Writes a TCF file containing the user-asserted or computed (if requested) probability and toggle count of the pins in the design.

The `write_tcf` command writes out a compressed TCF file if you add the `.gz` extension to the file name.

#### **Options and Arguments**

<code>-boundary_only</code>	Writes out the switching activities of the primary inputs ports and the leaf sequential instance output pins.
<code>-computed</code>	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.
<code>-duration <i>simulation_period</i></code>	<p>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</p> <p>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.</p> <p><b>Note:</b> Do not choose the period too small, otherwise the toggle count will be rounded to 0, because only integer numbers are written out.</p> <p><i>Default:</i> 1e+9 ns</p>
<code><i>file</i></code>	Specifies the name of the file to which to write the probability values and toggle count values.
<code>-hierarchy</code>	Writes out a TCF file in hierarchical format. By default, the RC-LP engine writes out a flat TCF file.
<code>-include_hier_ports</code>	Includes the (computed) switching activities for the hierarchical output ports.

## Command Reference for Encounter RTL Compiler

### Low Power Synthesis

---

-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-assserted 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:	<a href="#"><u>read_tcf</u></a> on page 993
Affected by these attributes:	<a href="#"><u>lp asserted probability</u></a> <a href="#"><u>lp asserted toggle rate</u></a>

**Command Reference for Encounter RTL Compiler**  
**Low Power Synthesis**

---

---

## Advanced Low Power Synthesis

---

- [apply\\_power\\_intent](#) on page 1016
- [check\\_cpf](#) on page 1019
- [check\\_library](#) on page 1022
- [commit\\_power\\_intent](#) on page 1028
- [create\\_library\\_domain](#) on page 1029
- [read\\_power\\_intent](#) on page 1030
- [report\\_low\\_power\\_cells](#) on page 1032
- [report\\_low\\_power\\_intent](#) on page 1033
- [verify\\_power\\_structure](#) on page 1034
- [write\\_power\\_intent](#) on page 1036

## apply\_power\_intent

```
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 <i>design</i>	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 <code>-module</code> option.
-module <i>module</i>	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 <code>-design</code> 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.

# Command Reference for Encounter RTL Compiler

## Advanced Low Power Synthesis

**-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:                      Suggestions:
-----
set_port_attributes             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:

<a href="#"><u>commit_power_intent</u></a> on page 1028
<a href="#"><u>read_power_intent</u></a> on page 1030
<a href="#"><u>write_power_intent</u></a> on page 1036

## 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.
<i>file</i>	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 <i>string</i>	Specifies the Conformal license to be used for this command.
-lp_only	Reports only the low power rule check errors and warnings.

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

---

	By default, non low-power related issues, such as structural issues are reported as well.
<code>-pre_exit string</code>	Specifies the name of the dofile (script) that must be sourced before the CLP exit command.
<code>-pre_read string</code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.
<code>-retention</code>	Reports only problems with the state retention rules.
<code>-run_dir directory</code>	Specifies the directory in which the required files for Encounter Conformal Low Power must be stored. <i>Default:</i> .clp
<code>-tclmode</code>	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.
0
```

## 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*

## **Command Reference for Encounter RTL Compiler**

### Advanced Low Power Synthesis

---

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 lists 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

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

---

**-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.

**-state\_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
=====
```

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

---

Library Domain	Total cells	LS cell	ISO cell	Combo (LS+ISO)	SR Flops
lib_074v(0.74)	345	2	25	2	5
lib_120v(1.2)	341	0	13	0	5
lib_090v(0.9)	345	2	25	2	5
lib_110v(1.1)	343	2	17	2	5

---

Unusable libcells

---

Library Domain	Total cells	LS cell	ISO cell	Combo (LS+ISO)	SR Flops
lib_074v(0.74)	249	34	45	16	49
lib_120v(1.2)	232	15	47	8	49
lib_090v(0.9)	249	34	45	16	49
lib_110v(1.1)	239	22	47	10	49

---

- The following command checks the libraries in library domain lib\_120v.

```
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:           enclosed
=====
```

---

Library Domain	Total cells	LS cell	ISO cell	Combo (LS+ISO)	SR Flops
lib_120v(1.2)	11	3	10	2	0

---

Unusable libcells

---

Library Domain	Total cells	LS cell	ISO cell	Combo (LS+ISO)	SR Flops
lib_120v(1.2)	0	0	0	0	0

---

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

- 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)  
-----  
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 0.74-0.74 down to  
.....PPPAD 0.74-0.74 1.2-1.2 up to  
.....PPPAD 0.74-0.74 1.2-1.2 up to  
lib_120v enable_high_out_high .....BH1XH 0.9-0.9 1.2-1.2 up to  
enable_low_out_low .....1CLXH 0.9-0.9 1.2-1.2 up to
-----
```

**Note:** If the libraries would have unusable isolation cells, the report would list the names of the isolation cells per library or library domain.

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

- The following command checks the function of library cell LSHLL1CLXL.

```
rc:/> check_library -libcell LSHLL1CLXL
=====
 Library      Libcell      Level   Isolation   Retention   Always   Function
 domain       domain       shifter    cell        flop        ON
 -----
 lib_074v     LSHLL1CLXL  true     true       false      false    Y = A * B
```

- The following command checks the level shifter characteristics of the specified cell.

```
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
```

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

- 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_SDFPHQNX10     HD65_LS_SDFPSQNX20
-----
Usable state-retention flops
-----

=====

Library Domain          Flops
-----
090_lib      HD65_LS_SDFNRX10_SRPG  HD65_LS_SDFNRX5_SRPG
110_lib      HD65_LS_SDFPHQNX10_SRPG  HD65_LS_SDFPHQX10_SRPG  HD65_LS_SDFPQNX10_SRPG
              HD65_LS_SDFPQX10_SRPG  HD65_LS_SDFPRQNX10_SRPG  HD65_LS_SDFPRQX10_SRPG
              HD65_LS_SDFPSQX10_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 1030

## **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 <i>design</i>	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:	<a href="#"><u>apply_power_intent</u></a> on page 1016 <a href="#"><u>read_power_intent</u></a> on page 1030 <a href="#"><u>write_power_intent</u></a> on page 1036
-------------------	---

## **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**

<i>domain_list</i>	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**

<code>[-1801   -cpf]</code>	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.  <i>Default:</i> -1801
<code>file</code>	Specifies the name of the power intent file. The file can have any name, suffix, or length.
<code>-module module</code>	Specifies the top module for which the power intent is read.  This option is required in the following cases: <ul style="list-style-type: none"><li>■ The file is read in before the design is read in.</li><li>■ The file is read after the design is loaded, but multiple designs were loaded in the session.</li></ul> By default, the tool assumes that the power content applies to the (single) design that was read in.
<code>-version string</code>	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 <code>upf_version</code> command.

### **Related Information**

[Using CPF for Multiple Supply Voltage Designs](#)

[Using CPF for Designs Using Power Shutoff Methodology](#)

## **Command Reference for Encounter RTL Compiler**

### Advanced Low Power Synthesis

---

[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 1016

[commit power intent](#) on page 1028

[write power intent](#) on page 1036

**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**

<code>-all</code>	Reports all violations.
<code>-continue_on_error</code>	Allows the tool to continue when low power rule check errors are encountered.
<code>-detail</code>	Provides a detailed report.
<code>-debug</code>	Creates a dofile and other required files for Encounter Conformal Low Power allowing you to debug any errors further in Conformal Low Power.
<code>file</code>	Redirects the report to the specified file.
<code>-isolation</code>	Reports only isolation related violations.
<code>-level_shifter</code>	Reports only level-shifter related violations.
<code>-license <i>string</i></code>	Specifies the Conformal license to be used for this command.
<code>-lp_only</code>	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.
<code>-post_synthesis</code>	Runs Conformal Low Power, after synthesis, on low power cells.
<code>-pre_exit <i>string</i></code>	Specifies the name of the dofile (script) that must be sourced before the <code>exit</code> command.
<code>-pre_read <i>string</i></code>	Specifies the name of dofile (script) that must be sourced before the libraries and the design are read.

## Command Reference for Encounter RTL Compiler

### Advanced Low Power Synthesis

---

-pre_synthesis	Runs Conformal Low Power, before synthesis, to check the existing low power cells.
-retention	Reports only state retention related violations.
-run_dir <i>directory</i>	Specifies the directory in which the required files for Encounter Conformal Low Power must be stored. <i>Default:</i> .
-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 1028

Affected by this attribute: [wclp lib statetable](#)

## **write\_power\_intent**

```
write_power_intent [-1801 | -cpf]
    [-basename string] [-design design]
    [-overwrite] [-to_macro]
```

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 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

`-basename string` Specifies the path and basename 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.

`-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](#)

[Using 1801 for Designs Using Multiple Supply Voltages and Power Shutoff Methodology](#)

Related commands:

[apply\\_power\\_intent](#) on page 1016

[commit\\_power\\_intent](#) on page 1028

[read\\_power\\_intent](#) on page 1030

**Command Reference for Encounter RTL Compiler**  
Advanced Low Power Synthesis

---

---

## Design Manipulation

---

- [change\\_link](#) on page 1041
- [change\\_names](#) on page 1043
- [clock\\_gating](#) on page 1050
- [delete\\_unloaded\\_undriven](#) on page 1051
- [edit\\_netlist](#) on page 1052
- [edit\\_netlist\\_bitblast\\_all\\_ports](#) on page 1054
- [edit\\_netlist\\_bitblast\\_port](#) on page 1055
- [edit\\_netlist\\_connect](#) on page 1056
- [edit\\_netlist\\_dedicate\\_subdesign](#) on page 1058
- [edit\\_netlist\\_delete](#) on page 1059
- [edit\\_netlist\\_disconnect](#) on page 1060
- [edit\\_netlist\\_group](#) on page 1062
- [edit\\_netlist\\_hier\\_connect](#) on page 1064
- [edit\\_netlist\\_new\\_design](#) on page 1065
- [edit\\_netlist\\_new\\_instance](#) on page 1066
- [edit\\_netlist\\_new\\_port\\_bus](#) on page 1068
- [edit\\_netlist\\_new\\_primitive](#) on page 1069
- [edit\\_netlist\\_new\\_subport\\_bus](#)
- [edit\\_netlist\\_ungroup](#) on page 1072
- [edit\\_netlist\\_uniquify](#) on page 1073
- [group](#)
- [insert\\_tiehilo\\_cells](#) on page 1075

## **Command Reference for Encounter RTL Compiler**

### Design Manipulation

---

- [mv on page 1078](#)
- [remove\\_cdn\\_loop\\_breaker on page 1080](#)
- [reset\\_design on page 1082](#)
- [rm on page 1083](#)
- [ungroup on page 1084](#)
- [uniquify](#)

## **change\_link**

```
change_link -instances instance_list
  { -design_name {instance | subdesign | design}
  | -libcell libcell }
  [-pin_map string] [-lenient]
  [-no_name_change]
  [-change_in_non_uniq_subdesign]
  [-copy_attributes] [-retain_exceptions]
```

Changes the reference of a hierarchical instance to the specified subdesign or design. The command also supports libcell to libcell reference changes as well as a hierarchical instance to libcell changes.

### **Options and Arguments**

**-change\_in\_non\_uniq\_subdesign**

Changes the link of the instance(s) in all instantiations of a non-uniquified subdesign.

**-copy\_attributes** Copies the attributes of the original leaf instance(s) to the new leaf instances.

**-design\_name {*instance* | *subdesign* | *design*}**

Specifies the design, subdesign, or hierarchical instance to which the link has to be changed.

**-instances *instance\_list***

Specifies the instance(s) whose reference has to be changed.

**-lenient** Leaves the pins floating if a pin map is not found.

**-libcell *libcell*** Specifies the library cell to which the instance link has to be changed.

**-no\_name\_change** Prevents renaming of the subdesign name.

**-pin\_map *string*** Specifies, as a Tcl list of lists, the required pin mapping for swapping.

**-retain\_exceptions** Keeps the original exceptions of the instance after replacing it with another link.

## Examples

- The following command changes the reference of the hierarchical instances `top/A`, `top/B`, and `top/C` to `patch`.

```
change_link -instances {/designs/top/instances_hier/A \
/designs/top/instances_hier/B /designs/top/instances_hier/C} \
-design_name /designs/patch
```

- The following example changes the reference of the hierarchical instance `add_0` to the design `add`:

```
rc:/> change_link -design_name add \
-instance /designs/test/instances_hier/add_0

CHLINK 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 unqualified. The following command changes a leaf instance in all instances of subdesign `A`.

```
change_link -instances {/designs/top/instances_hier/A1/instances_comb/g1} \
-libcell buf1 -change_in_non_uniq_subd
```

This command changes not only leaf instance `A1/g1` but also `A2/g1` and `A3/g1` with `buf1`.

**Note:** If you omit the `-change_in_non_uniq_subd` option, the tool will issue an error.

- The following command replaces hierarchical instance `A1` with `patch` and tries to retain all exceptions of `A1`.

```
change_link -instances {/designs/top/instances_hier/A1} \
-design_name /designs/patch -retain_exceptions
```

Exceptions defined for instance `A1` are copied to `patch` if the object for which the exception was originally defined is also found in `patch`. For example, an exception defined on `A1/d_reg` will be retained if `d_reg` also exists in `patch`.

- The following command specifies how to map the pins of the hierarchical instance `A` to the pins of design `new`. Instance `A` has pins `a`, `b`, `c`, and `d`. Design `new` has pins `e`, `f`, `g`, and `h`.

```
change_link -instances /designs/top/instances_hier/A \
-design_name /designs/new -pin_map {{a e} {b f} {c g} {d h}}
```

- The following commands replace hierarchical instance `U1` with top-level design `digit` and hierarchical instance `U2` with top-level design `digit_1` while retaining the original top-level design names as the subdesign names.

```
change_link -instances /des*/top/instances_hier/U1 -design_name /designs/digit
change_link -instances /des*/top/instances_hier/U2 -design_name /des*/digit_1
```

## **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 or 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.

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

**-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

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

`-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".

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

-restricted <i>string</i>	Specifies the list of strings that cannot be used in a name. Separate independent patterns to be replaced with a “space.” These strings are replaced with the -replace_str string.
-skip_bus_net	Skips nets that are part of busses are changed. Only scalar nets can be changed.
-subdesign	Changes the names of subdesign (object).
-subport_bus	Changes the names of subport bus objects.
	<b>Note:</b> You cannot change the left bracket, “[”, and the right bracket, “]”, because they are a part of the bus name when referencing individual bits of the bus.
-suffix	Adds a suffix to the names of the objects to be changed.
-system_verilog	Replaces SystemVerilog reserved words.
-uppertolower	Changes the names of all objects from uppercase to lowercase. This applies to objects of type instance, port, net, subport, design, and subdesign.
-verilog	Replaces Verilog reserved words.
-vhdl	Replaces VHDL reserved words as well as strings that start with a digit, an underscore, continuous (two or more) underscores, and end with an underscore. You do not need to specify the -case_insensitive, -instance, or -subdesign option if you specify the -vhdl option.

## Examples

- The following example adds a suffix \_t to the design name:

```
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
```

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

- 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  
.          SI2      clk1      in1[0]     in2[0]     in2[3]     in3[2]     in3[5]  
rc:/designs/test/ports_in> change_names -port_bus -map {"[" "(" ")" "]" "\")"}  
rc:/designs/test/ports_in> ls  
.          SI2      clk1      in1[0]     in2[0]     in2[3]     in3[2]     in3[5]
```

- The following two commands both change the brackets ("[" and "]") in the instance name to "x"s:

```
change_names -instance -restricted {[ ]} -replace_str "x"  
change_names -instance -restricted "\[ \]" -replace_str "x"
```

**Note:** There is no need to escape special characters when enclosing them in braces ({}). If you added the escape character inside the braces, the tool would try to replace this character as well with "x".

- The following example allows all capital and lower case letters, numbers, underscores, backslashes, and brackets:

```
rc:/> change_names -allowed \  
ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_\[\]
```

**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
```

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

- The following example indicates that the `d` cannot be used as the first character in a net name and that when a name collision would occur, the prefix `xx` can be used.

```
change_names -name_collision -first_res d -prefix xx
```

- The following example shows a part of a netlist that includes VHDL reserved words:

```
generate u1(.A (eg1[0] ), .B (B[0]), .Y (Y[0]));  
open u2(.A (eg1[1] ), .B (B[1]), .Y (Y[1]));  
endmodule u3(.A (eg1[2] ), .B (B[2]), .Y (Y[2]));
```

To change, or eliminate, the names of the VHDL reserved words, use the `-vhdl` option.

```
rc:/> change_names -vhdl
```

Now the netlist does not contain any VHDL keywords:

```
generate_cn u1(.A (eg1[0] ), .B (B[0]), .Y (Y[0]));  
open_cn u2(.A (eg1[1] ), .B (B[1]), .Y (Y[1]));  
endmodule u3(.A (eg1[2] ), .B (B[2]), .Y (Y[2]));
```

- The following example changes all the object names from lowercase to uppercase:

```
rc:/> ls /designs/test/ports_in  
./ in1[0] in1[1] in1[2] in1[3] in2[0] in2[1] in2[2]  
rc:/> change_names -lowertoupper  
Setting in1[3] --> IN1[3]  
Setting in1[2] --> IN1[2]  
Setting in1[1] --> IN1[1]  
...  
rc:/> ls /designs/test/ports_in  
./ IN1[0] IN1[1] IN1[2] IN1[3] IN2[0] IN2[1]
```

Notice how the lowercase design name change to uppercase as well. The `-uppertolower` option works similarly, except that it changes all uppercase letters to lowercase.

- The following example specifies the character ranges that are allowed when renaming instances.

```
change_names -regexp -allowed "a-zA-Z0-9" -instance
```

- The following example specifies that inside hierarchical instance `m2`, any changed names cannot end with the character `2`.

```
change_names -last_restricted 2 /designs/m1/instances_hier/m2
```

- The following example uses the `MY_UNCONN_` prefix for dummy nets when writing out the netlist.

```
write_hdl  
change_names -dummy_net_prefix MY_UNCONN_  
write_hdl
```

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

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 278

## **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**

<i>-all</i>	Additionally, deletes unloaded and undriven top-level ports from the design.
<i>design</i>	Specifies the design from which to remove the unused ports or subports.
<i>-disconnect</i>	Only disconnects the subports and hierarchical pins that are connected to constants and that do not fanout to anything.
<i>file</i>	Specifies the name of the file to which the output of the command should be redirected.
<i>-force_bit_blast</i>	Bitblasts modules that have individual bus bits that are unused or connected to constants and deletes the unused bus bits.
<i>-verbose</i>	Enables verbose output.

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

#### **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 1054

[edit\\_netlist bitblast\\_port](#) on page 1055

## **Command Reference for Encounter RTL Compiler**

### Design Manipulation

---

[edit netlist connect](#) on page 1056  
[edit netlist dedicate subdesign](#) on page 1058  
[edit netlist delete](#) on page 1059  
[edit netlist disconnect](#) on page 1060  
[edit netlist group](#) on page 1062  
[edit netlist hier connect](#) on page 1064  
[edit netlist new design](#) on page 1065  
[edit netlist new instance](#) on page 1066  
[edit netlist new port bus](#) on page 1068  
[edit netlist new primitive](#) on page 1069  
[edit netlist new subport bus](#) on page 1071  
[edit netlist uniquify](#) on page 1073  
[edit netlist group](#) on page 1062  
[edit netlist ungroup](#) on page 1072  
[ui respects preserve](#)

Affected by this attribute:

## **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:

```
%s_%d
```

### **Options and Arguments**

<i>design</i>	Specifies the design to which the port_bus belongs.
<i>instance</i>	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**

<i>constant</i>	Specifies the name of the constant to connect.
<code>-net_name string</code>	Specifies the user-defined name of the net.
<i>pin</i>	Specifies the name of an instance pin to connect.
<i>pgpin</i>	Specifies the name of the power or ground pin to connect.

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

*port*

Specifies the name of a design port to connect.

*subport*

Specifies the name of a subport (port on a hierarchical instance) to connect.

#### Example

- In the following example, A and B are already connected and C and D are already connected. When you connect A and C, the result is a net connecting A, B, C, and D.

```
rc:/designs/alu/ports_in> edit_netlist connect A C  
/designs/alu/nets/A_
```

#### Related Information

Related command:

[edit\\_netlist disconnect on page 1060](#)

**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**

<i>pin</i>	Specifies the name of an instance pin to disconnect.
<i>pgpin</i>	Specifies the name of the power or ground pin to disconnect.
<i>port</i>	Specifies the name of a design port to disconnect.
<i>subport</i>	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.

## **Command Reference for Encounter RTL Compiler**

### Design Manipulation

---

#### **Related Information**

[Using the edit\\_netlist Command in Design for Test in Encounter RTL Compiler](#)

Related command:      [edit\\_netlist connect](#) on page 1056

## **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.
```

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

#### Related Information

[Grouping and Ungrouping Objects](#) in *Using Encounter RTL Compiler*

Related command: [edit\\_netlist ungroup](#) on page 1072

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 1056

## **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**

<code>-name <i>string</i></code>	Specifies the name of the new design.
<code>-quiet</code>	Suppresses the warning messages regarding naming conflicts.

### **Examples**

- The following example creates a new design called DESIGNA.

```
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.

```
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 1068

## **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**

<i>-name string</i>	Specifies the name of the new instance.
{ <i>design subdesign libcell</i> }	Specifies the object to instantiate.
<i>-quiet</i>	Suppresses the warning messages regarding naming conflicts.
{ <i>subdesign design</i> }	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
rc:/> ls /designs/TEST/instances_hier/
      /designs/TEST/instances_hier/TEST_SUB
```

- The following example tries to create a new subdesign called TEST\_SUB under the TEST design. However, a subdesign by that name already exists. In such cases, a number will be appended to the end of the specified name and an error message indicating the naming conflict will be printed. The following example specifies the -quiet option to suppress this warning.

```
rc:/> edit_netlist new_instance -name TEST_SUB /designs/TEST -quiet \
      /designs/TEST
      /designs/TEST/instances_hier/TEST_SUB3
```

The name given to the new subdesign in this case is TEST\_SUB3.

## **Command Reference for Encounter RTL Compiler**

### Design Manipulation

---

#### **Related Information**

Related command: [edit\\_netlist new\\_subport\\_bus](#) on page 1071

## **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**

<i>design</i>	Specifies the name of the design for which to create the <code>port_bus</code> .  The design name can be omitted if there is only one top-level design.
<code>-input</code>	Creates an input <code>port_bus</code> .
<code>-input -output</code>	Creates a bidirectional <code>port_bus</code> .
<code>-left_bit <i>integer</i></code>	Specifies the leftmost bit index of the bus.
<code>-name <i>string</i></code>	Specifies the name of the new <code>port_bus</code> .
<code>-output</code>	Creates an output <code>port_bus</code> .
<code>-right_bit <i>integer</i></code>	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 1065

## **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.

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

- 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**

<i>design</i>	Specifies the name of the instance for which to create the <code>subport_bus</code> .
<code>-input</code>	Creates an <code>input</code> <code>subport_bus</code> .
<code>-input -output</code>	Creates a <code>bidirectional</code> <code>subport_bus</code> .
<code>-left_bit <i>integer</i></code>	Specifies the leftmost bit index of the <code>subport_bus</code> .
<code>-name <i>string</i></code>	Specifies the name of the <code>new</code> <code>subport_bus</code> .
<code>-output</code>	Creates an <code>output</code> <code>subport_bus</code> .
<code>-right_bit <i>integer</i></code>	Specifies the rightmost bit index of the <code>subport_bus</code> .

### **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 1066

## **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**

<i>instance</i>	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.
<code>-prefix</code>	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 1062

## **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.

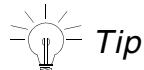
Alias for [edit\\_netlist group](#).

## 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 high and tie low cells 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 unqualified 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 be used for 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.  <i>Default:</i> 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.

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

**-hilo libcell | -hi libcell -lo libcell**

Specifies a tie hi/lo cell to replace constants. 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.
tiehi_cell is /libraries/slow/libcells/TIEHI , tiehi_cell is /libraries/slow/
libcells/TIEHI

Info : 2 scan pins which are loads of '0' in /designs/m1 are skipped.
Use the '-all' option to avoid skipping of scan pins.
/designs/m1/instances_seq/foo/bx_reg/pins_in/SE
/designs/m1/instances_seq/tm_reg/pins_in/SE

Done connecting 1'b0 and 1'b1 to TIELO/TIEHI cells
```

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

- The following example shows two modules, U1 and top. When the `insert_tiehilo_cells -skip_unused_hier_pins` command is used, the pin B of the instantiation of U1 in module top will be skipped.

```
module U1(A, B, C, Sel, Z);
    input A, B, C;
    input [1:0] Sel;
    output Z;
    wire A, B, C;
    wire [1:0] Sel;
    wire Z;
    wire n_0, n_1, n_2, n_3, n_4, n_5;
    NAND2X1 g27(.A (n_5), .B (n_4), .Y (Z));
    NAND2X1 g28(.A (n_3), .B (n_2), .Y (n_5));
    NAND2X1 g29(.A (n_1), .B (A), .Y (n_4));
    NOR2X1 g30(.A (n_0), .B (Sel[0]), .Y (n_3));
    INVX1 g31(.A (n_1), .Y (n_2));
    NOR2X1 g32(.A (Sel[1]), .B (Sel[0]), .Y (n_1));
    INVX1 g33(.A (C), .Y (n_0));
endmodule

module top(a, b, c, sel, z);
    input a, b, c;
    input [1:0] sel;
    output z;
    wire a, b, c;
    wire [1:0] sel;
    wire z;
    U1 inst_U1(.A (a), .B (1'b0), .C (c), .Sel (sel), .Z (z));
endmodule
```

### Related Information

#### [Removing Assign Statements in Using Encounter RTL Compiler](#)

Related attributes:

[ui respects preserve](#)  
[use tiehilo for const](#)

## Command Reference for Encounter RTL Compiler

### Design Manipulation

---

#### **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
- support\_bus
- test\_clock
- test\_clock\_domain
- test\_signal

#### **Options and Arguments**

<code>-flexible</code>	Indicates to be flexible for renaming when there is a collision.
<code>new_name</code>	Specifies the new name for the specified object.
<code>object</code>	Specifies the object to rename.
<code>-slash_ok</code>	Indicates that the destination name can have embedded slashes.

## Examples

- The following example changes the name of design `comp` to `comp_test`.

```
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:

```
rc:/> mv /designs/dpldalgns/subdesigns/mux muxYYY  
rc:/> ls /designs/dpldalgns/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:

```
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 1043

# Command Reference for Encounter RTL Compiler

## Design Manipulation

## **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 10/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
```

## **Command Reference for Encounter RTL Compiler**

### Design Manipulation

---

#### **Related Information**

Related command: [report cdn\\_loop\\_breaker](#) on page 432

## **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**

<i>design</i>	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
./
```

## rm

```
rm object... [-quiet]
```

Removes an object from the design hierarchy. This command is similar to its UNIX counterpart.

For a current list of the objects that can be removed, refer to the command help.

If the hierarchical pin or port bus object has a net connection, the net is disconnected first and then the object is removed.

If you remove a design, the CPF-related information will also be removed from the design hierarchy.

**Note:** The `rm` command does not work on the `pin` or `port` object.

### Options and Arguments

<i>object</i>	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:

```
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.

```
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 <i>instances</i>	Specifies a list of instances that should not be ungrouped.
-flatten	Recursively ungroups all the specified instances.
-force	Forces to ungroup all hierarchies. <b>Note:</b> Before mapping, this will include non-user hierarchies.
<i>instance</i>	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. <b>Note:</b> 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 <i>string</i>	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 <i>integer</i>	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 1072

## **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 1088
- [define\\_attribute](#) on page 1089
- [mesg\\_make](#) on page 1094
- [mesg\\_send](#) on page 1096
- [parse\\_options](#) on page 1097

## **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.

## Command Reference for Encounter RTL Compiler

### Customization

---

**-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.

**-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)

## 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  
==>     }  
==>     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
```

## Command Reference for Encounter RTL Compiler Customization

---

The following command is a valid use of the newly created `test_check` attribute:

```
rc:/> set_attribute test_check 1 /  
Setting attribute of root '/' : 'test_check' = 1
```

The following command would be an invalid use:

```
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):

```
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:

```
rc:/> set_attribute test_set 1  
Setting attribute of root '/' : 'test_set' = 1  
rc:/> get_attribute test_set * -h
```

Usage: `get_attribute <string> [<object>+]`

```
<string>:  
      attribute name  
[<object>+]:  
      object of interest (must be unique)  
  
attribute category: test  
  
attribute name: test_set  
      category: test ()  
      object type: root  
      access type: read-write  
      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`:

```
rc:/> set_attribute test_set 2 /  
Setting attribute of root '/' : 'test_set' = 2
```

## Command Reference for Encounter RTL Compiler

### Customization

---

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  
  
/object_types/root/attributes/test_compute
```

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**

<code>-error</code>	Creates an error message.
<code>-group <i>string</i></code>	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.  <b>Note:</b> If you want to create a message for an internal group, you must specify an existing group name.
<code>-id <i>integer</i></code>	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.  <i>Default:</i> 1
<code>-info_priority <i>integer</i></code>	Creates an info message with the specified priority. You can specify a number between 2 and 8.
<code>-internal_group</code>	Specifies whether the message belongs to an internal group.
<code>-long_description <i>string</i></code>	Defines the help of the message.
<code>-short_description <i>string</i></code>	Specifies the title or the description of the message.
<code>-warning</code>	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 1096

## **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**

<code>-caller string</code>	Specifies the name of the calling procedure.
<code>-file_info string</code>	Specifies to which file the message applies.
<code>message</code>	Identifies the message to be sent. The identification is in the form of <i>group-id</i> . Refer to <code>mesg_make</code> for an explanation of <i>group</i> and <i>id</i>
<code>-newline</code>	Prints a new (empty) line before the message
<code>-object_info string</code>	Prints the object type and
<code>string</code>	Describes the context-specific help.

### **Examples**

- The following example accesses the message created in [Example](#) on page 1095 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 1094

## **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:

```
proc your_procedure {args} {  
    ... (code that implements the procedure)  
}
```

## Command Reference for Encounter RTL Compiler

### Customization

---

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>(<dirtytypes>) ? <help>"
```

The question marks in the above string mean that these fields of the string are optional.

You cannot specify multiple string values if you are specifying a flag name. That is, if you are specifying a flag, you cannot also specify the som (string, optional, multiple values) or srm (string, required, multiple values) combinations.

<name> is the name of the flag. For example, -number.

<x> is a single character indicating the type of the option:

b	Boolean
d	directory object
e	enum
f	float
n	integer number
s	string

<y> is a single character indicating whether the option is optional or required

o	optional
r	required
x	obsolete

<z> is a single character indicating whether lists are accepted

m	Accepts multiple values: lists OK
s	Accepts single value only: no lists

*<dirtytypes>* 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 \
```

## Command Reference for Encounter RTL Compiler Customization

---

```
==> "-level nos The level to work on"    level \
==> "dos(subdesign|instance) An object to work on"    object
1
rc:/> puts $object
/designs/alu/subdesigns/addinc65
rc:/> puts $top
1
rc:/> puts $level
300
```

- The following example shows how online help works:

```
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
```

# A

---

## Applets

---

- [Introduction](#) on page 1102
- [applet](#) on page 1103
- [applet avail](#) on page 1104
- [applet install](#) on page 1106
- [applet list](#) on page 1107
- [applet load](#) on page 1108
- [applet update](#) on page 1109
- [applet version](#) on page 1110
- [applet whatis](#) on page 1111

## 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.

## Command Reference for Encounter RTL Compiler

### Applets

---

#### **applet**

```
applet {avail | install | list | load  
        | 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.
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.

## Command Reference for Encounter RTL Compiler

### Applets

---

#### **applet avail**

```
applet avail
    [-location applet_installation]
    [-outdated] [-local]
```

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**

**-location *applet\_installation***

Specifies the location of the local applet installation from where applets can be loaded.

Specify this option if you want to override the value of the `applet_search_path` root attribute.

By default, the location specified through the `applet_search_path` attribute is used.

**-local**

Only reports information for applets that exist in the local installation(s).

**-outdated**

Only reports scripts for which a newer version is available on the server. New applets not present in the local installation are also listed.

#### **Examples**

- The following command lists all available applets.

```
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)

  Name      | Installed | Server Ver. | Summary Description
  -----+-----+-----+-----+
  compare_power |       6 |     1.15 | Generate HTML comparison report of power metrics..
  create_tcse |      5 |     1.42 | generate / load / save testcases
  hermes |      3 |     1.14 | Hermes Design Assistant
  lock |      5 |     1.12 | Guarantee atomic operations using a LOCKFILE...
  manual_assert |      3 |      1.3 | Apply default toggle rate to all synthesis ...
```

```
NOTE: To install an applet, use 'applet install -location <applets directory> <applet name>'
rc:/>
```

## Command Reference for Encounter RTL Compiler Applets

---

- The following command lists the outdated applets.

```
rc:/> applet avail -outdated
=====
...
=====

Info: Collecting applet server information...
#####
# 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>'
rc:/>
```

- The following command lists the local applets.

```
rc:/> applet avail -local
=====
...
=====

Info: Collecting applet server information...
#####
# Local/Server information:
#   Local Install (/home/me/.localApps/rc)
#   Remote Server (<install>/tools.<platform>/lib/applets)

Name      | Installed | Server Ver. | Summary Description
-----+-----+-----+
compare_power |       6 |      1.13 | Generate HTML comparison report of power metrics..
create_tcase |      5 |      1.42 | generate / load / save testcases
hermes |      3 |      1.14 | Hermes Design Assistant
manual_assert |      3 |      1.3 | Apply default toggle rate to all synthesis ...

NOTE: To install an applet, use 'applet install -location <applets directory> <applet name>'
rc:/>
```

#### **applet install**

```
applet install
    [-location applet_installation]
    [applet_list] [-force]
```

Installs one or more applets and their corresponding dependencies.



An applet is not available for execution until it is loaded.

#### **Options and Arguments**

<i>applet_list</i>	Only installs the specified applets. By default, all applets on the server will be installed.
<i>-force</i>	Indicates to overwrite the installed applets. By default, the installed applets are not overwritten.
<i>-location applet_installation</i>	Specifies the location where to install the local applets. Specify this option if you want to override the value of the <i>applet_search_path</i> root attribute. By default, the location specified through the <i>applet_search_path</i> 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
```

Displays all applets that have been loaded and that are available for use in the current session

### **Example**

The following example shows the applets that were loaded.

```
rc:/> applet load compare_power
...
rc:/> applet list
=====
...
=====

The following applets have been loaded:
    compare_power
    generate_report
    time_info
```

```
rc:/>
```

## Command Reference for Encounter RTL Compiler

### Applets

---

#### applet load

```
applet load  
  [-location applet_installation]  
  [applet_list]
```

Loads one or more applets and their corresponding dependencies.



An applet is not available for execution until it is loaded.

#### Options and Arguments

*-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.

*applet\_list*

Only loads the specified applets.

**By default, all applets on the server will be loaded.**

#### Example

The following command loads the *compare\_power* applet together with all its dependencies.

```
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/generate_report/  
generate_report.tcl' (Tue Aug 23 15:52:34 -0700 2011)...  
Sourcing '/install_path/tools.lnx86/lib/applets/time_info/time_info.tcl' (Tue  
Aug 23 15:52:34 -0700 2011)...  
Loading applet 'compare_power'...  
rc:/>
```

## **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/myname/.localApps/rc...
Updating applet 'time_info' (1.1 -> 1.2)...
Info: Connecting to server....
Info: Collecting applet server information...
```

## **applet version**

applet version

Provides version information of the applet command

### **Example**

The following command shows the current version of the applet command.

```
rc:>/ applet version  
applet command Version: 4
```

## Command Reference for Encounter RTL Compiler

### Applets

---

#### **applet whatis**

```
applet whatis [-detail] applet_list
```

Provides a summary description of the capabilities of the specified applets.

#### **Options and Arguments**

<i>applet_list</i>	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}
=====
disclaimer info...
=====

Info: Collecting applet server information...
Applet: time_info
    Location: install_path/lib/applets
    Version: 2
    Summary: Create time stamp and keep track of runtime metrics

Applet: lock
    Location: install_path/lib/applets
    Version: 2
    Summary: Guarantee atomic operations using a LOCKFILE, wrapper
functions
```

- The following command requests detailed information for the `time_info` applet.

```
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.
```

## **Command Reference for Encounter RTL Compiler**

### **Applets**

---

# Index

---

## A

abstract model  
  logic, writing [278](#)  
  scan, reading [871](#)  
  scan, writing [914](#)  
adding  
  design [1065](#)  
  directory to stack [60](#)  
  help for user command [1088](#)  
  user-defined command [1097](#)  
  user-defined message [1094](#)  
area, reporting [427](#)  
assigning, paths to cost group [348](#)  
ATPG file, writing out [642, 833, 900](#)  
Attribute  
  defining new  
    defining custom [1089](#)  
attributes  
  retrieving value [84](#)  
  setting value [111](#)

## B

bitblasting  
  one port [1052, 1055](#)  
bitblasting, all ports [1054](#)  
body segment, defining [739](#)  
boundary scan logic  
  previewing changes [789](#)  
busses  
  renaming [1043](#)

## C

cells  
  reporting cell count [427](#)  
  reporting technology library cells  
    used [471](#)  
changing  
  attribute value  
    defined by RC [111](#)  
  directory in design hierarchy [39](#)  
  names

  of instances of an object type [1043](#)  
path constraints [338](#)  
  UNIX working directory [90](#)  
channel masking types [637](#)  
clock gating  
  insertion in mapped netlist [977](#)  
clock inputs, reporting [411](#)  
clock waveform, defining [320](#)  
clock-gating logic  
  editing [971](#)  
  inserting and connecting observability  
    logic [978](#)  
  merging instances [974](#)  
  removing [982](#)  
  reporting [434](#)  
  test input, connecting [973](#)  
color  
  highlighting [137](#)  
  label [163](#)  
commands, alphabetical list of [21](#)  
compatible test clocks, declaring [892](#)  
concatenating  
  scan chains [675](#)  
configuring, pads for DFT [677](#)  
congestion map  
  create snapshot for documentation [168](#)  
connecting  
  pins, ports [1056](#)  
  scan chains [681](#)  
  test input of clock-gating logic [973](#)  
constraints  
  checking [543](#)  
  reading in [229](#)  
cost group  
  assigning paths to [348](#)  
  defining [325](#)  
creating  
  binding for Chipware component [188](#)  
  Chipware component [190](#)  
  delay constraint for specific path [342](#)  
  design [1065](#)  
  design representation in design  
    hierarchy [362](#)  
  HDL library [194](#)  
  hierarchy in design [1062, 1074](#)  
  implementation for Chipware

# Command Reference for Encounter RTL Compiler

- 
- component [192](#)
  - instance of [1066](#)
  - library domains (MSV) [1029](#)
  - parameter for Chipware component [198](#)
  - pin [200](#)
  - port bus [1068](#)
  - subdesign [1058](#)
  - subport bus [1071](#)
  - synthetic operator [195](#)
  - timing constraints for instance [326](#)
  - user-defined message [1094](#)
  - critical path, reporting timing slack for [541](#)
  - current directory
    - in design hierarchy [61](#)
    - in UNIX [101](#)
  - D**
    - defining
      - boundary scan segment [695](#)
      - clock [320](#)
      - configuration mode [699](#)
      - cost group [325](#)
      - DFT test clock [719, 761](#)
      - input and output delays [328](#)
      - multi-cycle path [334](#)
      - paths for timing constraints [353](#)
      - scan clock for LSSD [745, 748](#)
      - shift-enable signal for DFT [751](#)
      - test-mode signal for DFT [765](#)
      - user\_defined scan chain [739](#)
      - user-defined abstract segment [689](#)
    - design
      - area [427](#)
      - creating
        - generic netlist [381](#)
        - new design [1065](#)
      - incremental optimization [382](#)
      - instantiating [1066](#)
      - mapping [380](#)
      - removing [1078](#)
      - renaming [1043](#)
      - synthesizing [379](#)
      - uniquifying [1073, 1086](#)
    - design hierarchy
      - adding hierarchy level [1062, 1074](#)
      - changing directories in [39](#)
      - finding object type [46](#)
      - listing information [52, 53](#)
    - removing hierarchy level [1072](#)
    - removing objects [1052, 1059, 1083](#)
    - renaming objects [1078](#)
    - returning current position [61](#)
    - design rule constraints
      - reporting violations on [448](#)
    - DFT clock domain
      - associating with scan chain [740](#)
      - specifying compatible test clocks [892](#)
    - DFT MBIST clocks
      - defining [719](#)
    - DFT rule violations
      - checking flip-flops for [648](#)
      - fixing [771](#)
    - DFT rules, list of rules checked [648](#)
    - DFT test clocks
      - declaring as compatible [892](#)
      - defining [761](#)
    - directories
      - changing in
        - design hierarchy [39](#)
        - UNIX [90](#)
      - listing contents in
        - design hierarchy [52, 53](#)
        - UNIX directory [98](#)
      - returning current position in
        - design hierarchy [61](#)
        - UNIX [101](#)
    - directory stack
      - adding new directory [60](#)
      - displaying directory list [42](#)
      - tracing previous directory [58](#)
    - disconnecting pins, ports [1060](#)
    - displaying
      - current position in design hierarchy [61](#)
      - directory stack [42](#)
      - UNIX directory, contents [98](#)
  - E**
    - executing
      - script [89](#)
      - UNIX shell command [113, 114](#)
    - exiting, from RTL Compiler [83](#)
  - F**
    - fanin, reporting [415](#)
    - fanout, reporting [419](#)

# Command Reference for Encounter RTL Compiler

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 [65](#)  
fixing, DFT rule violations [771](#)  
flattening, instances [1084](#)  
flip-flops  
  checking for DFT rule violations [648](#)  
  reporting scannability [459](#)  
Floorplan (DEF)  
  Reading [602](#)  
  Writing [624](#)

## G

generic netlist  
  creating [381](#)  
GUI  
  update [131](#)

## H

HDL files, reading [218](#)  
HDL Viewer  
  remove data [133](#)  
head segment, defining [741](#)  
help  
  adding for new command [1088](#)  
  providing for commands [71, 88](#)  
highlight  
  clear [136, 150](#)  
  color [137, 159](#)

## I

incremental optimization [382](#)  
input delay, defining [328](#)  
inserting  
  clock-gating logic in mapped netlist [977](#)  
  shadow logic [843](#)  
  test point [849](#)  
  user-defined test point [855](#)  
  wrapper cell [857](#)  
instances

grouping [1062, 1074](#)  
removing from design [1078](#)  
renaming [1043](#)  
ungrouping [1072](#)  
isolation rule  
  removing [1078](#)

## L

leakage power, reporting [507](#)  
length  
  abstract segment [690](#)  
  maximum chain length [742](#)  
level-shifter groups  
  removing [1078](#)  
library cell, instantiating [1066](#)  
library domains  
  creating [1029](#)  
licenses  
  checked out, listing [96](#)  
  checking out [93](#)  
listing  
  object information [52, 53](#)  
lockup element type, chain-specific [743](#)  
logic abstract model, definition [278](#)  
logic abstract, writing [278](#)  
loop breakers  
  removing [1080](#)  
  reporting [432](#)

## M

mapped netlist  
  creating [382](#)  
  writing out [278](#)  
mapping  
  design [379](#)  
  non-scan flops with scan-equivalent [880](#)  
memory resources, reporting [487](#)  
messages  
  creating customized [1094](#)  
  reporting [489](#)  
  sending [1096](#)  
  suppressing [117](#)  
mode, specifying for power, timing analysis, and optimization [317](#)  
MSV design  
  library domains, creating [1029](#)

# Command Reference for Encounter RTL Compiler

multi-cycle path, defining [334](#)

## N

names  
  adding prefix [1045](#)  
  limiting length [1045](#)  
  mapping characters [1045](#)  
net power, reporting [507](#)  
nets  
  renaming [1043](#)

## O

object  
  finding base name [38](#)  
  finding directory name [41](#)  
object path  
  return a selection list for instance, net,  
    pin, and port objects [130](#)  
object type, finding [46](#)  
object types  
  listing all attributes for a type [84](#), [111](#)  
objects  
  connecting [1056](#)  
  disconnecting [1060](#)  
  filtering on attribute value [43](#)  
  instantiating [1066](#)  
  listing information for an object [52](#), [53](#)  
  removing from design hierarchy [1052](#),  
    [1059](#), [1083](#)  
  renaming [1078](#)  
  returning type of an object [65](#)  
  selecting in design hierarchy [43](#)  
observability logic for clock-gating logic  
  inserting [978](#)  
OPCG logic  
  previewing changes [823](#)  
OPCG segments  
  previewing connections [679](#)  
optimizing  
  for area or timing [373](#)  
  incrementally [379](#)  
  RTL [380](#)  
output delay, defining [328](#)  
output, redirecting [106](#)

## P

parameterized module, elaborating [362](#)  
path  
  return path for a selected object [130](#)  
path constraints, modifying [338](#)  
paths  
  assigning to cost group [348](#)  
  creating for timing exception [353](#)  
  unconstraining [345](#)  
pins  
  connecting manually [1056](#)  
  disconnecting [1060](#)  
port buses  
  creating new object [1068](#)  
  removing from design [1078](#)  
  renaming [1043](#)  
ports  
  bitblasting [1054](#)  
  connecting manually [1057](#)  
  disconnecting [1060](#)  
  renaming [1043](#)  
power  
  reporting [507](#)  
  sorting report [509](#)  
power domain  
  removing [1078](#)  
primitive cell, creating [1069](#)  
probability values of nets  
  reading or updating [993](#)  
  writing to SAIF file [1010](#)  
  writing to TCF file [1012](#)

## R

reading  
  HDL files [218](#)  
  SAIF file [988](#)  
  SDC constraints [229](#)  
  TCF file [993](#)  
remove  
  data [133](#)  
removing  
  clock-gating logic [982](#)  
  directory from stack [58](#)  
  hierarchy from design [1072](#)  
  object from design hierarchy [1059](#),  
    [1083](#)  
renaming

# Command Reference for Encounter RTL Compiler

---

instances of object type [1043](#)  
object [1078](#)

reporting  
area of design [427](#)  
cell types, used [471](#)  
clock information of design [440](#)  
clock-gating information [434](#)  
design rule violations [448](#)  
DFT registers [459](#)  
DFT setup information [463](#)  
DFT violations [467](#)  
inferred datapath operators [443](#)  
instance information [477](#)  
memory resources [487](#)  
messages in current session [489](#)  
net information [499](#)  
port information [505](#)  
scan chains [449](#)  
timing information [541](#)

retrieving  
attribute value  
defined by RC [84](#)  
clock ports [411](#)  
fanin information [415](#)  
fanout information [419](#)  
input ports [73](#)  
object information [52, 53](#)  
output ports [74](#)  
UNIX working directory [101](#)

RTL Compiler  
exiting [83](#)  
quit [105](#)  
resuming process [118](#)  
suspending process [118](#)

RTL optimization [380](#)

RTL power analysis, enabling [509](#)

**S**

SAIF file  
reading [988](#)  
writing [1008](#)

scan abstract model, creating [914](#)

scan abstract model, reading [871](#)

scan chain  
tool-created  
test clock domain associated with  
name [704, 706, 736, 754, 779](#)

scan-chain configuration  
previewing [684](#)

reporting [449](#)

scan chains  
connecting [681](#)  
defining [739](#)  
mixing edges of same clock on [892](#)  
removing [1078](#)  
reporting [449](#)

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, 767, 799](#)

scan segment  
defining [689, 704, 706, 736, 754, 779](#)  
removing [1078](#)  
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 [964](#)

scripts  
executing [89](#)  
writing out [291](#)

SDC constraints  
reading in [229](#)  
unsupported constructs [229](#)  
writing out [294](#)

shadow logic  
inserting [843](#)  
previewing changes [845](#)

shift-enable port  
for scan segment [691](#)  
confirming if connected [690, 737](#)  
for shift-enable signal [731, 753](#)

shift-enable signal  
defining [751, 765](#)  
specifying default [751](#)  
specifying for chain [743](#)

slack, reporting at timing endpoints [542](#)

SPEF  
Reading [608](#)  
Writing [626](#)

subdesigns  
creating by copying existing  
subdesign [1058](#)

## Command Reference for Encounter RTL Compiler

---

instantiating [1066](#)  
renaming [1043](#)  
uniquifying [1073, 1086](#)  
support buses  
  creating new object [1071](#)  
  removing from design [1078](#)  
  renaming [1043](#)  
supports  
  connecting manually [1057](#)  
  disconnecting [1060](#)  
switching power, reporting [507](#)  
synthesis  
  creating generic netlist [381](#)  
  incremental optimization [382](#)  
  mapping [382](#)

## T

tail segment, defining [743](#)  
TCF file  
  reading [993](#)  
  writing [1012](#)  
test mode signal, defining [765](#)  
test point  
  inserting [849](#)  
  possible types [851](#)  
timing constraints  
  deriving for instance [326](#)  
  reporting violations on [541](#)  
  writing out [291](#)  
timing exception, created by  
  modifying path constraints [338](#)  
  overriding default clock edge  
    relationship [334](#)  
  specifying timing constraints [342](#)  
  unconstraining paths [345](#)  
timing units [541](#)  
toggle counts of nets  
  reading or updating [993](#)  
  writing to SAIF file [1010](#)  
  writing to TCF file [1012](#)

## U

uniquifying, design or subdesign [1073,](#)  
[1086](#)  
UNIX shell command  
  executing in RTL Compiler [113, 114](#)  
user-defined test point, inserting [855](#)

utilization map  
  create snapshot for documentation [168](#)

## W

wireload model, reporting [427](#)  
worst paths, reporting timing for [544](#)  
wrapper cell, inserting [857](#)  
writing  
  compression model [906](#)  
  DFT abstract [914](#)  
  logic abstract [278](#)  
  netlist file [278](#)  
  SAIF file [1008, 1010](#)  
  scanDEF file [964](#)  
  SDC constraints [294](#)  
  TCF file [1012](#)  
  timing and design rule constraints [291](#)